LOT SIZING MODELS FOR GROUP TECHNOLOGY PRODUCTION SYSTEMS

A Thesis Submitted
In Partial Fulfilment of the Requirements
for the Degree of
MASTER OF TECHNOLOGY

By
V. JACOB NEAL

to the
NDUSTRIAL AND MANAGEMENT ENGINEERING PROGRAMME
INDIAN INSTITUTE OF TECHNOLOGY, KANPUR
APRIL, 1985

CERTIFICATE

This is to certify that this work entitled, Lot Sizing Models for Group Technology Production Systems, by Mr. V.Jacob Neal, has been carried out under my supervision and that it has not been submitted elsewhere for the award of a degree.

[illegible]

(J.L. Batra)
Professor and Head
Industrial and Management Engg.
Indian Institute of Technology
Kanpur -- 208 016

April, 1985

IMEP-1985-M-NEA-LOT

ACKNOWLEDGEMENTS

It is with immense pleasure and great respect that I express my deepest sense of gratitude to Dr. J.L. Batra for his invaluable guidance and encouragement throughout my thesis work.

I am thankful to my friends, especially Kasi, Krishna Prasad, Naren and Kiran who made my stay at IIT pleasant, memorable and enjoyable.

I also express my whole hearted thanks to all members of IME family for their constant inspiration and ever-helping gesture.

V. JACOB NEAL

CONTENTS

ABSTRACT

In this thesis we consider the lot sizing of components in a Group Technology (GT) production system. The manufacture of components in the GT production environment involves the establishment of cells comprising of several machines. Each cell is capable of handling a part family and the components move through the various machines in the cell as far as possible sequentially. In the present work we assume that it is possible to obtain a part family and a production cell so that no back tracking of components is required. Further, it is assumed that each stage comprises of one machine. A review of the literature suggested that in such a production environment, work-in-process (WIP) inventory contributes significantly to the total cost of production/inventory system. In the present work we have developed four mathematical models, for lot sizing of components belonging to a part family which is to be manufactured in a GT cell.

The mathematical models developed assume a constant demand rate for the components, and no inter-cell movements for all components.

The first model basically is an extension of the Wilson's model for determining the economic production quantity. The various machines are identified as stages and a component belonging to the part family is manufactured sequentially on the various stages. The model assumes that the lot (economic production quantity) will move from one stage to the next stage only after the entire lot has been manufactured on the present stage. It is shown that WIP inventory influences the economic lot size as well as the total cost of the production/ inventory system.

The second model considers the splitting of the lot at a stage into batches to reduce the contribution of the WIP inventory. Each batch after it is manufactured at a given stage is transported to the next stage for further processing. The model is named as constant lot size model with lot splitting. A two stage algorithm involving a heuristic procedure and an optimisation procedure has been suggested for this model.

The third model called the variable lot size model assumes the Crowston integrality Theorem, on the lot sizes at various stages. Crowston's integrality theorem states that if i denotes any stage, a(i) denotes its successor stage and N denotes the final stage then there exists a set of optimal lot sizes $\{Q_1, Q_2, \ldots, Q_N\}$ such that for all $i < N$ the ratios $K_i = Q_i/Q_{a(i)}$ are positive integers. A dynamic programming algorithm has been suggested for solving the variable lot size model.

In the last model, we combine the important features of second and third models, viz., the splitting of the lot at a stage and assigning variable lot sizes for the lots to be produced at various stages. This model is referred as the variable lot size model with lot splitting. A heuristic solution methodology has been suggested for this model.

The solution methodology of the second model is explained through an illustrative example. The proposed algorithms are coded in Fortran - 10 for implementation on DEC 1090 system. The computational performances of the various algorithms have been investigated for problems of varied sizes.

Problem size varied from single stage to 10 stages. The input parameters for various problems were generated randomly in the specified ranges. The constant lot size model with lot splitting is compared with variable lot size model with lot splitting.

CHAPTER I

INTRODUCTION

1.1 GROUP TECHNOLOGY CONCEPT:

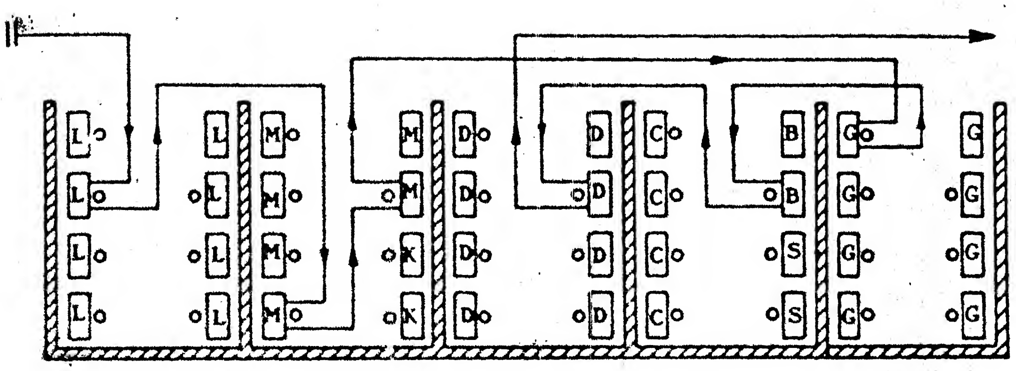
Group Technology (GT) is a very progressive method of organising production and especially it is becoming popular in those industries which are engaged in medium and small batch production. The principles of mass production applied in batch production would result in lot of duplication of paper work and loss of machining time. One fundamental step capable of bringing the mass production principles within the reach of medium batch production is the adaptation of group machining approach. Professor Mitrafanov (1) is the pioneer in the field of group production.

Group Technology is a technique of identifying and bringing together related or similar components in a production process in order to take advantage of their similarities in design, dimensions, geometrical shapes, raw material and tooling, by making use of the inherent economies of flow production methods. The aim is to substantially reduce the set-up times and to improve the delivery performance by reducing the throughput times. This is achieved by organising a large number of diverse components into families which require similar manufacturing processes and providing the most suitable manufacturing facilities for groups of families by designing cells having machine centres exclusively for processing the group of components. The physical nature of difference between traditional batch production and group production is shown in Fig. 1.1.

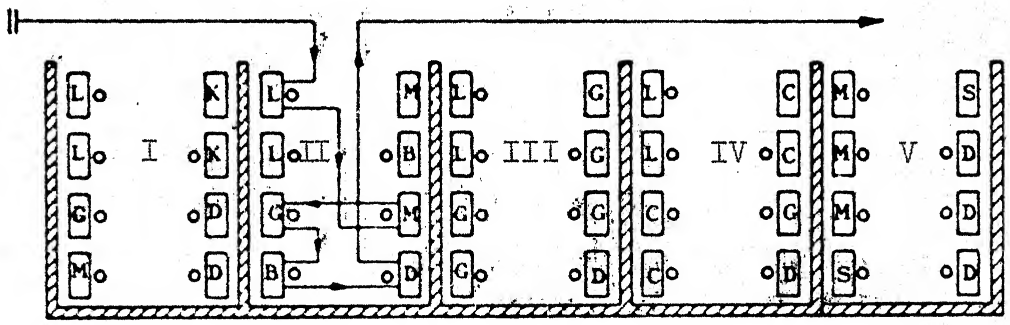
The introduction of group concepts in production organisation has two main origins. Firstly methods were developed by engineers who were mainly interested in finding the techniques which would reduce the set-up time resulting in increased capacity and better utilisation of equipment. Secondly, behavioural scientists advocated the use of group concepts in organisational design for increasing the workers motivation and job satisfaction. The models based on group concepts developed by the engineers and behavioural scientists have number of similarities in terms of structure and solution methodologies and have been implemented successfully in varied situations.

Burbidge (2) has listed number of successful applications of group technology in industry. Maximum benefits of GT applications result in a production environment involving a wide variety, medium volume of production of components, requiring different types of production facilities.

The following benefits result from the introduction of Group Technology.



(a) Functional Organisation



(b) Group Technology Organization

Fig. 1.1: Difference between traditional batch production and group production.

1. For each component all the required operations are done inside one cell. This results in the reduced throughput time.

2. It is possible to sequence the loading operations well in advance since the components (parts) included in the family are prespecified. Thus it becomes possible to obtain a better load balance for the production cells and scheduling of components on machines within the cell.

3. By the application of composite component principle tooling, jigs and fixtures can be standardised, there by reducing the set-up times considerably resulting in increased available capacity.

4. Duplication in design and process planning effort can be reduced due to simplification in variety of design and use of schemes of component classification and coding.

5. The responsibility for quality can be assigned as all the operations are performed in the cell. Hence quality levels improve for all components.

6. GT simplifies the production control, material flow system and material handling.

7. Apart from operational benefits, the social and psychological benefits include better motivation, higher job satisfaction, improved skills and better quality of life.

1.2 DESIGN OF GT PRODUCTION SYSTEM:

The design of GT production system embraces several functions such as component classification and coding, establishment of groups and cells, development of a group layout and determination of lot sizes, sequences, schedules for components and load on cells.

The following are the main characteristics of a GT production system.

1. GT production system comprises of machining cells.
2. Each component is classified and coded into family of components.
3. Each cell contains all machines and equipment needed to complete all the operations for the component families assigned to the cell. This is to reduce the inter cell movements of components.
4. Each component has definite sequence of operations and the components move through the production cell with minimum back tracking.

The design and implementation of a GT production system involves the consideration of the following problems.

1. Component classification and coding schemes.
2. Design of part families and machine cells.

3. Sequencing and scheduling of components belonging to part families assigned to the cell.
4. Lot sizing of components.

1.2.1 Component Classification and Coding:

The objective of classification and coding schemes is to classify the components by their features so that components having similar code numbers possess similar features. The three basic component features used for classification are shape, function and manufacturing operations and tooling. Different classification schemes use different combinations of these features. Some of the important classification and coding schemes are optiz system, VUOSCO system, PERA system and Brisch system (3). The problem with most of the schemes available in the literature is that they only consider the design aspects of the component and do not account for certain important aspects like the machines used, the total requirement of component etc. Therefore, there is need to develop classification and coding schemes for GT production system based on features pertaining to production, design and resources used. Such classification and coding schemes would help in process planning, production control, data processing etc. of a GT production system.

1.2.2 Design of Part Families and Machine Cells:

The design of groups and cells involves the identification of components to be made and machines to be installed in each cell. Quantitative methods such as production flow analysis (PFA) or component flow analysis (CFA) can be used to establish the groups of components and machine cells. Analytical methods such as clustering technique, graph-theoretic approach can also be used to identify the component machine cells. These methods identify component machine groups by analysing the machine to machine routes followed by all components.

A layout for the machines in the cell has to established so as to machine all the components in the part family with minimum back tracking.

1.2.3 Sequencing and Scheduling of Part Families:

Having determined what is to be manufactured in each cell, the sequence of operations for the components must be decided. This may be arranged to achieve a particular aim: maximum labour or machine utilisation, minimum throughput time, minimum setup time by sequencing components by similarity of tooling requirement, minimum material consumption, or an optimum combinations of these several objectives.

Ideally, with GT the groups should receive a series of orders at regular period intervals. Under reasonable assumptions such as all the components are processed in the cell, an extension of Johnson's algorithm can be used for a two stage problem to minimise the make span time. Similarly for a N-stage problem a branch and bound procedure can be used to minimise the make span time. The sequencing and scheduling decisions should incorporate the latest information in giving a solution. Also the supervisors should be given sufficient freedom to over-ride any solution as they can get the work done more efficiently.

Loading on various machines and the cells should be uniform throughout the plant to avoid any misunderstanding among the management, workers and unions.

1.2.4 Lot Sizing of Components:

After determining the sequence of operations for each component, then it is necessary to decide for every component how much to produce at each production facility. This quantity is known as lot size produced at the facility in a single setup.

1.3 MOTIVATION AND SCOPE FOR PRESENT STUDY:

One of the important problems in the manufacturing systems based on GT concepts is the determination of economic lot sizes for the various stages (machines) comprising the cell. The determination of lot sizes in GT manufacturing environment is some what different from the traditional manufacturing system for the following reasons. In a GT production system, one has greater control over the movement of components compared to the traditional methods of production (3). Fig. 1.2 illustrates the differences in production through-put time for the traditional scheme of manufacturing and GT production system. The elements of throughput time in a functional production system are queueing time, machining time and transportation time. In GT production system the transportation time is completely eliminated because all machines needed for processing the entire part family are located within the cell. The queueing time is also minimum due to the greater autonomy given to the cell. So the throughput time for a given component of a part family mainly constitute the total time spent by the component in the GT cell. The total time spent by the component has the following two elements.

1. Total setup time i.e. the setup time required on all the machines for the production of the given lot of components.
2. Total machining time for processing the entire lot of the component on the various machines in the cell.

The total setup time is constant for the component irrespective of the lot size. However, the total machining time of the lot would depend on the lot size and influence the throughput time for the lot. Fig. 1.3 shows the relationship between throughput time and lot size. The work-in-process (WIP) inventory increases with an increase in the throughput time which in turn is a function of the lot size. WIP inventory

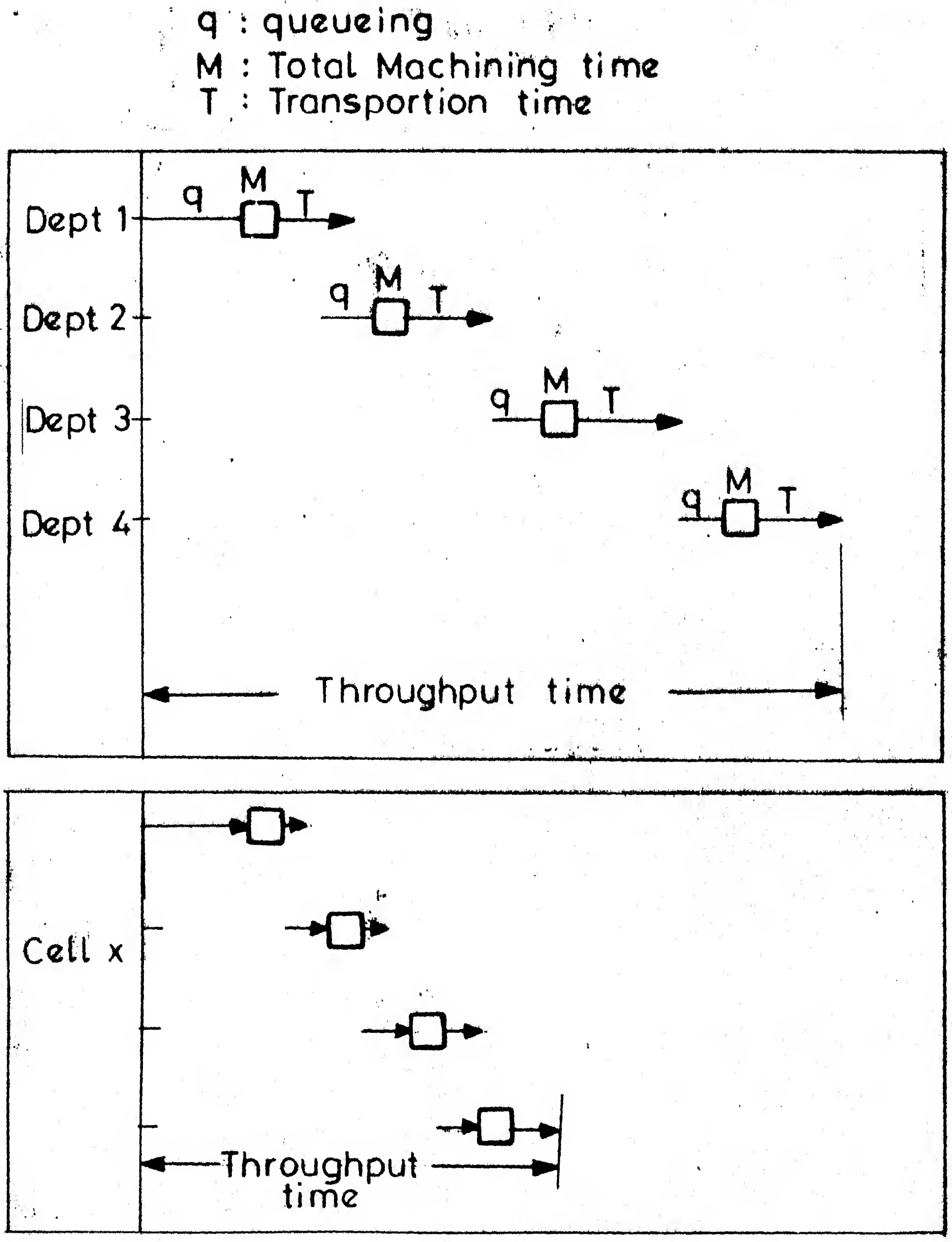


FIG. 1·2 DIFFERENCES IN PRODUCTION THROUGHPUT TIME OF FUNCTIONAL PRODUCTION AND GT.

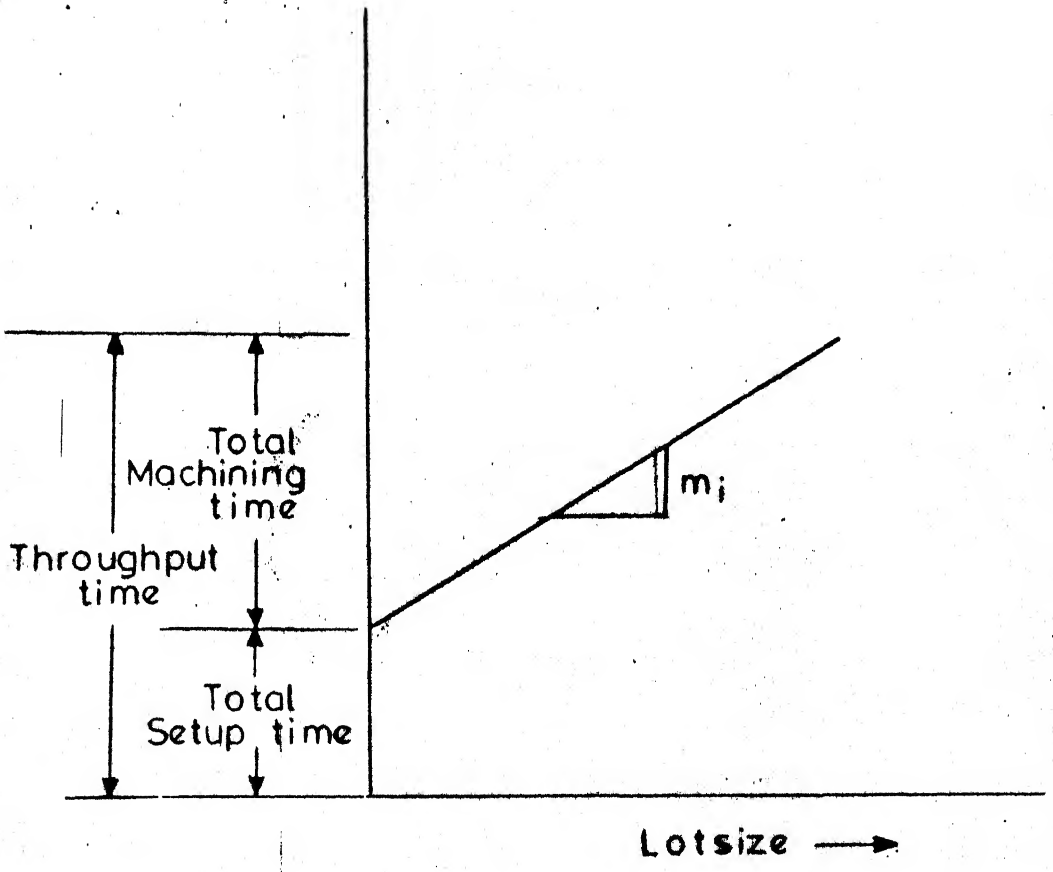


FIG. 1·3 RELATIONSHIP BETWEEN LOTSIZE AND THROUGHPUT TIME IN GT. PRODUCTION SYSTEM cost is a major contributor in the total cost of production inventory system operating in a GT environment (4). One of the ways to reduce the contribution of work-in-process inventory towards the total cost would be through the splitting of the lot to be manufactured at the machine/stage into batches and transporting the batch to the next stage. With this particular idea in view several lot sizing models have been developed in the present thesis.

The first model which is an extension of the basic Wilson's economic production quantity model has been developed to reflect the influence of WIP characterised in cell type of manufacture.

Since the GT production system permits the transportation of the lot to the next stage in the cell in no time and at lower costs, the lot can be split into batches while processing, and can be transported to the next stage, for further processing so as to reduce the work-in-process inventory. Under such an assumption, a constant lot size model with lot splitting is developed. A two stage algorithm, involving a heuristic procedure and an optimisation procedure, is presented.

In a GT cell there can be machines with high production rates involving high setup costs. To distribute the high costs associated with the setup, it may become necessary to produce lots of different sizes at various stages. For such a situation a variable lot sizing model assuming integer ratios of lot-sizes at successive stages is developed. The model is formulated using dynamic programming (DP) and the recurrence relations are developed to solve the problem.

Finally, the features of constant lot size model and variable lot size model are combined and another model called variable lot size model with lot splitting is developed. A heuristic procedure is suggested to solve the problem.

The solution methodologies of the various models have been coded in Fortran 10 for DEC-1090 system and were tested for randomly generated problems with input variations in a specified range. The computational experience with each model is reported.

1.4 ORGANISATION OF THE THESIS:

Chapter II deals with a brief literature review on group technology with special reference to GT lot sizing. In Chapter III, four models viz., an extension of Wilson's EPQ model, a constant lot size model with lot splitting, a variable lot size model, and a variable lot size model with lot splitting, are presented. For every model, a brief statement of the problem is presented first and is followed by formulation and solution methodology. The solution methodology for constant lot size model with lot splitting, is explained using an illustrative example.

Further, computational experience based on solving a set of randomly generated problems of varied sizes is presented for the last 3 models. In Chapter IV, conclusions based on the present study along with suggestions for further work are presented.

CHAPTER II

LITERATURE SURVEY

In this chapter a brief review of the literature on Group Technology with special reference to the lot sizing problem is presented. Waghodekar and Sahu (30) have given an excellent bibliography of more than 450 papers on the subject. We present a review of the important literature on GT under the following categories.

1. Design of cells and groups,
2. Group scheduling and sequencing,
3. Lot sizing problem,
4. Performance evaluation of GT production systems.

2.1 DESIGN OF CELLS AND GROUPS:

The following approaches have been reported for the design of cells and groups.

1. Rule of thumb approach by Edwards (5)
2. Composite component approach by Edwards (5)
3. Classification and coding schemes by Burbidge (2).
4. Flow analysis
   i) Production flow analysis by Burbidge (2),
   ii) Component flow analysis by El-Essaway (6),

5. Approaches using similarity co-efficients,
   i) Cluster Analysis by McAuley (7),
   ii) Graph theoretic approach by Rajagopalan and Batra (8),
6. Cell formation using Monte Carlo simulation by Crookall and Baldwin (9),
7. Mathematical classification by Purcheck (10),
8. Matrix clustering technique by Mc Cormick (11).

The first four approaches are some what qualitative in nature while the remaining are analytical approaches. A detailed discussion on these approaches is given by Wagho-dekar and Sahu (12).

2.2 GROUP SCHEDULING AND SEQUENCING:

Petrov (13) has developed four inter related scheduling models for different types of route sequences and component flows. Hitomi and Ham (14) have suggested a technique for scheduling multi product multi-stage manufacturing systems using Ignall and Shrage branch and bound approach. Hitomi, Ham and Yoshida (15) considered group scheduling decisions under due date constraints. However all these models assume that the set-up time is included in the processing time. Kishore (16) separated the setup time from processing time and developed an extension of Johnson's algorithm for the two stage problem considering the criterion of minimising the make span time. For a N-stage problem, a branch and bound procedure and a heuristic procedure has been developed.

In addition to group scheduling, machine loading and product mix decisions represent major problem areas for group production planning and scheduling. Hitomi and Ham (17) have considered problems from the view point of GT, for a single stage production. Agrawal (18) has extended the work of Hitomi and Ham and suggested optimisation techniques for multi-stage problems.

2.3 GT LOT SIZING:

The lot sizing in GT is a special case of lot sizing in multi-stage production inventory system.

For a multi stage production system Crowston et al (19) have suggested that in an optimal schedule the lot size at any given stage should be an integer multiple of the lot size at its immediate predecessors and suggested that the problem can be solved by examining all combinations of such integer values.

Chakravarty (20) has considered the production planning and lot sizing problems, for mutually independent machine component groups. Assuming the integrality theorem of Crowston (19) and no splitting of the lot for inter-stage shipment, production cycle time of every machine was found considering the set-up and inventory costs. Also a network based design approach to integrate the lot sizing and layout decisions has been presented.

Ignall and Veinott (21) have suggested system myopic policies for multi-stage production system under continuous review with constant demand over infinite planning horizon. System myopic policies optimise a given objective function with respect to any two stages and ignore the multi-stage interaction effects.

Wagner and Whitin (22) have developed a dynamic lot size model to solve the lot sizing problem of single product with known demand in discrete time periods. Zangwill (23) has suggested a network formulation for determining dynamic economic lot sizes with back logging. The network formulations facilitate the development of efficient dynamic programming algorithms for obtaining the optimal dynamic lot sizes.

Goyal (24) developed a mathematical model for lot size scheduling on a single machine for stochastic demand. A method for computing the lot size in each time period is presented. Newson (25) developed a network based heuristics to solve the capacitated lot size problems with fixed resources and variable resources.

2.4 PERFORMANCE MEASUREMENT BY SIMULATION:

Gupta and Tompkins (26) have studied the performance of a GT production system with a simulation model written in Simscript. The performance characteristics include average stay time, intercell and intra-cell movements, number of orders completed in time etc., Ang and Willey (27) have presented a simulation model which compares the Pure GT and hybrid GT. In hybrid GT, inter cell movements are permitted to certain extent while no inter-cell movements are permitted in pure GT production system.

CHAPTER III

LOT SIZING MODELS

In this chapter the following lot sizing models have been developed.

1. An extension of Wilson's EPQ model,
2. Constant lot size model with lot splitting,
3. Variable lot size model,
4. Variable lot size model with lot splitting.

For every model, a brief statement of the problem is presented first and is followed by assumptions, notations, formulation and solution methodology. The computational experience for models 2, 3 and 4 based on solving a set of randomly generated problems is also presented.

MODEL 1:

3.1 AN EXTENSION OF WILSON'S EPQ MODEL:

3.1.1 Statement of the Problem:

Consider a GT cell manufacturing a part family of components with known annual demand. The components are machined sequentially by the machines in the cell and the inter transfer time of components between machines is negligible. The problem is to determine the Economic Production Quantity (EPQ) for every component considering the costs due to work in process, setup and finished goods inventory.

3.1.2 Assumptions:

1. All components are processed in the cell and inter cell movements are not permitted.
2. Demand rate is constant over the time horizon.
3. The inter transfer time of components between machines is negligible.

3.1.3 Notation:

R	:	Cost of operating the cell/unit time.

For a given component j of a part family,

Q_j	:	Lot size,
D_j	:	Annual demand rate
M_j	:	Unit material cost,
V_j	:	Value added in the cell,
W_j	:	Work in process inventory value,
TS_j	:	Total setup time for all machines used by the component,
TM_j	:	Total machining time on all machines used by the component,
h_j	:	Average inventory carrying cost.

3.1.4 Model Formulation:

The throughput time per lot is given by $TGT = TS_j + TM_j.Q_j$. As the component progresses through the cell, value will be added to it. The value added can be expressed as,

$$V_j = \text{(Throughput time/unit) x Cost of operating the cell}$$

$$= \left(\frac{TS_j}{Q_j} + TM_j\right) R$$

The work in process inventory value per cycle can be given as,

$$\text{VWIP/cycle} = \text{(Unit Material Cost} + \tfrac{1}{2}\text{ value added) Lot Size}$$

$$= (M_j + \tfrac{1}{2} V_j)\ Q_j$$

The annual WIP inventory value can be written as,

$$A_j = \text{(VWIP/cycle) x No. of cycles x Throughput time/lot}$$

$$= (M_j + \tfrac{1}{2} V_j)\ Q_j \times \frac{D_j}{Q_j} \times (TS_j + TM_j\ Q_j)$$

Simplifying we get,

$$A_j = D_j(M_j + \tfrac{1}{2} V_j) \times (TS_j + TM_j\ Q_j)$$

The finished goods inventory value per unit may be written as

$$FG(Q_j) = M_j + V_j$$

The total cost is the sum of setup costs, finished goods inventory carrying costs and the WIP inventory carrying cost. The total cost function can be written as,

$$TC(Q_j) = \frac{F_j D_j}{Q_j} + h_j \frac{Q_j}{2} FG(Q_j) + h_j D_j (M_j + \frac{V_j}{2})$$
$$(TS_j + TM_j\ Q_j)$$

Substituting for $FG(Q_j)$ and V_j, we get,

$$TC(Q_j) = \frac{F_jD_j + h_jD_j\,TS_j^2\,R/2}{Q_j} + Q_j\left[\frac{1}{2}h_j(M_j + TM_j\,R) + h_jD_j\,TM_j\,\left(M_j + \frac{RTM_j}{2}\right)\right] + \frac{h_j\,TS_j\,R}{2} + h_jD_jM_j\,TS_j + h_jD_jR\,TS_j\,TM_j \qquad (3.1)$$

Since the function is a single variable convex function, differential calculus can be applied to solve for Q_j. Differentiating with respect to Q_j and solving it by equating to zero, we get,

$$Q_j^* = \sqrt{\frac{2F_j\,D_j + D_j\,h_j\,TS_j^2\,R}{h_j(M_j + TM_jR) + 2h_jD_jTM_j\,(M_j + (TM_jR)/2)}} \qquad (3.2)$$

Here Q_j^* is the optimal lot size for the component j of the part family.

Substituting $Q_j^* = Q_j$ in Eq. (3.1) we get the optimal total cost $TC(Q_j^*)$.

3.1.5 <u>Comparison with Basic Wilson's EPQ Model:</u>

For the Wilson's EPQ model, the total cost expression without considering cost of the work in process inventory value can be written as

$$TC(\bar{Q}_j) = \frac{F_j\,D_j}{\bar{Q}_j} + FG\,(\bar{Q}_j)\,h_j\,\frac{\bar{Q}_j}{2}$$

$$= \frac{F_j\,D_j}{\bar{Q}_j} + \frac{h_j\,\bar{Q}_j}{2}\left[M_j + \left(\frac{TS_j}{\bar{Q}_j} + TM_j\right)R\right] \qquad (3.3)$$

The economic production quantity $\bar{Q}_j$ can be obtained similarly as,

$$\bar{Q}_j^* = \sqrt{\frac{2F_j D_j}{h_j (M_j + TM_j R)}} \qquad (3.4)$$

Substituting $\bar{Q}_j^*$ in Eq. (3.3) the optimal cost $TC(\bar{Q}_j^*)$ can be obtained.

The two models are compared for variety of problems with variations of input parameters within a given range. The ranges selected for the input parameters are given in Table 3.1. The input data selected for a sample of 6 problems is given in Table 3.2. The economic production quantity and the total cost for the two models are presented in Table 3.3. For all the sample problems the consideration of WIP inventory results in smaller production lot size as well as total cost of the production inventory system.

MODEL 2:

3.2 CONSTANT LOT SIZE WITH LOT SPLITTING:

3.2.1 Statement of the Problem:

Consider a GT cell comprising of N stages (each stage corresponds to a machine) manufacturing a part family of components with known annual demand. The components are produced sequentially on the various stages, in the cell. A constant lot is to be produced on all stages. The lot being produced at a particular stage can be split into batches which

Table 3.1: Input Ranges

Variation in Demand (pieces)	Variation in Material Cost (Rs.)	Variation in Setup hrs.	Variation in Machining Time (minutes)
1000-10000	2.0-10.0	1.0-12.0	8-35

Table 3.2: Input Data

R = 75000 Rs./year, h_j = 10 percent
Production hours available for the cell/year = 2880

Problem No.	Demand D_j	Material Cost M_j (Rs.)	Total Setup Time TS_j (Hrs.)	Total Machining Time/Unit MS_j (Minutes)
1.	6000	2.00	9	20
2.	4000	4.00	12	25
3.	5000	5.00	8	30
4.	10000	3.0	11	35
5.	9000	2.0	10	23
6.	1000	4.0	6	20

Total 3.3: Comparison of Two Models.

Prob-lem No.	EPQ Model with WIP Inventory		Wilson's EPQ Model	
	Q_j^*	$TC(Q_j^*)$	$\bar{Q}_j^*$	$TC(\bar{Q}_j^*)$
1.	1264	2126.724	1616	2393.101
2.	985	1942.10	1297	2578.83
3.	883	1948.02	1075	2394.10
4.	964	5791.132	1769	6878.505
5.	1625	1450.866	2590	1810.26
6.	448	637.31	496	696.8392

can be transported to the next stage for further processing, even when processing of the remaining lot is still in progress at this particular stage. The inter transfer time between stages is negligible. The problem is to determine the constant lot size for all the stages and the number of batches for the production of the lot at each stage such that the total cost of the system arising on account of setup, transportation and inventory is minimised. The lot size, the batch size at each stage and the number of batches into which the lot is split must be integers.

3.2.2 Assumptions:

1. The demand rate is deterministic and constant over the planning horizon.
2. For all the stages the setup costs are fixed.
3. The inventory carrying costs are linear in nature.
4. The transportation cost per batch at a stage is independent of the number of units transported.
5. The inter transfer time of components between stages is negligible.
6. The production rate at each stage is greater than the demand rate.

3.2.3 Notation:

For a component j of a part family,

D : Annual demand rate of the component (final product)

Q : Constant lot size,

N : Number of stages,

For the component at any stage i,

x_i : Batch size,

y_i : Number of batches,

F_i : Setup cost per lot,

h_i : Unit inventory holding cost per unit time,

T_i : Transportation cost per batch,

m_i : Machining time,

E_i : Elapsed time between stages i and i+1,

$R^{\updownarrow}$: A real value R rounded to the nearest integer,

$R^{\uparrow}$: A real value R rounded to the higher integer,

$R^{\downarrow}$: A real value R rounded to the lower integer.

3.2.4 Model Formulation:

Fig. 3.1 represents the inventory building between stages i and i+1, for the case $m_i > m_{i+1}$. The first slanted line represents the cumulative production at stage i. The corners of various triangles formed with this line indicate the availability of batches for transportation to the next stage i+1. The second slanted line represents the cumulative production at stage i+1. The dotted lines crossing this line represent the depletion of stage i inventory. The trapezoid enclosed by solid lines represents the time weighted inventory at stage i. The inventory at stage i builds up over time period Qm_i during which y_i number of x_i sized batches are transported to stage i+1.

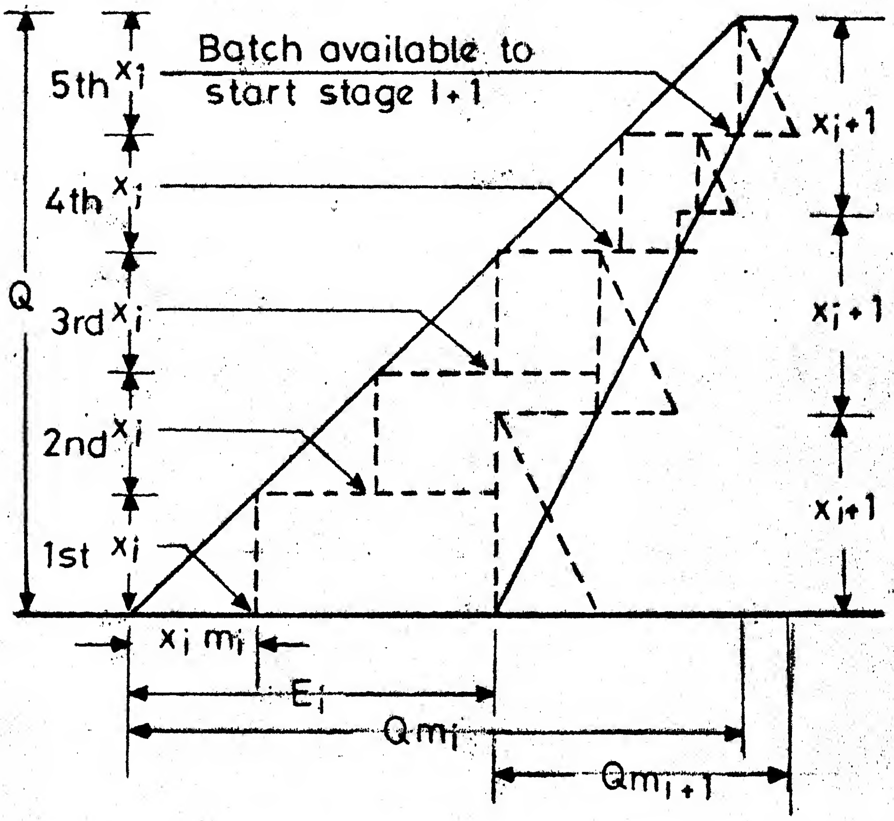


FIG. 3·1 TIME-WEIGHTED INVENTORY AT STAGE i WHEN $y = 5$; $y_{i+1} = 3$ AND $m_i > m_{i+1}$

From the Fig. 3.1, we observe that for uninterrupted production, the elapsed time between stages i and i+1 is given by,

$$E_i = Qm_i + x_i m_{i+1} - Qm_{i+1}$$

The time-weighted inventory at stage i is given by the area of the trapezoid,

$$\Delta = \frac{Q}{2} [(Qm_i + x_i m_{i+1} - Qm_{i+1}) + x_i m_{i+1}] \qquad \text{for } m_i > m_{i+1}$$

Similarly when $m_i \leq m_{i+1}$, it is easy to verify that,

$$\Delta = \frac{Q}{2} [(x_i m_i + (Qm_{i+1} + x_i m_i - Qm_i))]$$

The two expressions can be combined as,

$$\Delta = \frac{Q}{2} [2x_i \min (m_{i+1}, m_i) + Q|m_i - m_{i+1}|]$$

Substituting $x_i = Q/y_i$ and simplifying,

$$\Delta = \frac{Q^2}{2} [\frac{\min (m_{i+1}, m_i)}{y_i} + \frac{1}{2} |m_i - m_{i+1}|]$$

The average inventory cost can be calculated by multiplying the time weighted inventory with inventory carrying cost per unit time and the number of cycles. It is given by the following expression,

$$\sum_{i=1}^{N} h_i \frac{D}{Q} \Delta$$

Substituting for Δ, the expression for the average inventory cost is written as,

$$D \sum_{i=1}^{N} h_i \, Q \left[\frac{\min(m_{i+1}, m_i)}{y_i} + \frac{1}{2} |m_i - m_{i+1}| \right]$$

The total setup and transportation cost considering all the stages is given as,

$$S(F_i, T_i) = D \sum_{i=1}^{N} \left(\frac{F_i}{Q} + \frac{y_i T_i}{Q} \right)$$

The total cost function is obtained by summing up the setup costs, transportation cost and the average inventory costs. Thus the total cost of producing the component on the N-stages is given by,

$$TC(Q,Y) = D \sum_{i=1}^{N} \left[\left(\frac{F_i}{Q} + \frac{y_i T_i}{Q} \right) + h_i Q \left\{ \frac{\min(m_{i+1}, m_i)}{y_i} + \frac{1}{2} |m_i - m_{i+1}| \right\} \right]$$

where $Y = \{y_1, y_2, \ldots, y_n\}$

The problem of determining the optimum lot size can now be formulated as,

$$\text{Min } TC(Q,Y) = AQ + \frac{B}{Q} + \sum_{i=1}^{N} \left[\left(\frac{b_i}{Q} \right) y_i + \frac{a_i Q}{y_i} \right] \quad (3.5)$$

subject to,

y_i = Positive integer,

Q = Positive integer

x_i = Q/y_i positive integer for all i

$A, B, a_i, b_i > 0$ (3.6)

where,

$$A = D \sum_{i=1}^{N} \frac{h_i}{2} |m_i - m_{i+1}|$$

$$B = D \sum_{i=1}^{N} F_i$$

$$a_i = Dh_i \min (m_i, m_{i+1}) \quad \forall \ i$$

and,

$$b_i = DT_i, \quad \forall \ i$$

3.2.5 Solution Methodology:

The optimisation of the objective function given by (3.5) is carried out in two phases. In the first phase a heuristic solution to the problem is found. In the heuristic procedure, initially the values of Q, x_i and y_i are determined iteratively, maintaining integrality for each of them. In finding the optimal solution, the results of heuristic procedure are used as starting values for developing the upper and lower bounds on the lot size within which the optimal solution lies. Starting from the lower bound of the lot size the optimal solution is found by scanning several combinations of batch sizes and number of batches. An efficient scanning method is developed for this purpose.

3.2.6 Heuristic Procedure:

The objective function (3.5) can be rewritten as,

$$TC(Q,Y) = [A + \sum_{i=1}^{N} (a_i/y_i)]Q + [B + \sum_{i=1}^{N} b_i y_i]/Q \qquad (3.7)$$

Substituting $x_i = Q/y_i$ in (3.5) we obtain,

$$TC(Q, Y) = AQ + \frac{B}{Q} + \sum_{i=1}^{N} a_i x_i + \sum_{i=1}^{N} b_i/x_i$$

Let,

$$\theta(x_i) = a_i x_i + \frac{b_i}{x_i} \quad \text{and}$$

$$\phi(y_i) = (\frac{b_i}{Q})\, y_i + (a_i Q)/y_i$$

The total cost function given in (3.8) can now be rewritten as,

$$TC(Q, X) = AQ + \frac{B}{Q} + \sum_{i=1}^{N} \theta(x_i) \qquad (3.9)$$

$$TC(Q, Y) = AQ + \frac{B}{Q} + \sum_{i=1}^{N} \phi(y_i) \qquad (3.10)$$

where, $X = \{x_1, x_2, \ldots, x_n\}$

Lemma 1: For $P, q > 0$ the positive integer $\dot{K}$ that minimises the function $f(K) = PK + \frac{q}{K}$ is

$$\dot{K} = [\frac{q}{P} + 0.25]^{1/2}$$

Proof: The optimal $\dot{K}$ must satisfy,

$$f(\dot{K} - 1) \geq f(\dot{K}) \qquad (3.11)$$

and,

$$f(\dot{K}) \leq f(\dot{K} + 1) \qquad (3.12)$$

Using (3.11), we get,

$$P\dot{K} + \frac{q}{\dot{K}} \leq P(\dot{K} - 1) + \frac{q}{\dot{K}-1}$$

Rearranging,

$$\dot{K}(\dot{K}-1) \leq q/P$$

Adding 0.25 on both sides and simplifying,

$$(\dot{K} - 0.5)^{1/2} \leq \frac{q}{P} + 0.25$$

$$(\dot{K} - 0.5) \leq (\frac{q}{P} + 0.25)^{1/2} \qquad (3.13)$$

Similarly using (3.12) we get,

$$P\dot{K} + \frac{q}{\dot{K}} \leq P(\dot{K}+1) + \frac{q}{\dot{K}+1}$$

Rearranging,

$$\dot{K}(\dot{K} + 1) \geq q/P$$

Adding 0.25 on both sides and simplifying

$$(\dot{K} + 0.5)^2 \geq (q/P + 0.25)$$

$$(\dot{K} + 0.5) \geq (q/P + 0.25)^{1/2} \qquad (3.14)$$

Combining (3.13) and (3.14) we get,

$$(\dot{K} - 0.5) \leq (q/P + 0.25)^{1/2} \leq \dot{K} + 0.5$$

This implies that $\dot{K}$ is rounded to nearest integer of $(q/P + 0.25)^{1/2}$.

This proves the lemma.

Lemma 2: For $P, q > 0$ the minimum of $f(K) = PK + q/K$ is

$$K = (q/P)^{1/2} \quad \text{and}$$

$$f(K\ ) = 2(qP)^{1/2}$$

Proof: Since $f(K)$ is convex, for $K > 0$ solving $\frac{df(K)}{dK} = 0$, we get K and $f(K\ )$.

For the time being relaxing the constraint on y_i and using Lemma 1, we get from Eq. (3.9),

$$Q = (B/A + 0.25)^{1/2} \updownarrow \qquad (3.15)$$

$$x_i = (b_i/a_i + 0.25)^{1/2} \updownarrow \quad \forall\ i \qquad (3.16)$$

Substituting $Q = y_i x_i$ in (3.9) and again using Lemma 1 we get,

$$y_i = (\frac{1}{x_i^2}(B/A) + 0.25)^{1/2} \updownarrow \quad \forall\ i \qquad (3.17)$$

After establishing X,Y vector from (3.7), a lot size which is nearest multiple of all y_i s is obtained as,

$$Q' = [(B + \sum_{i=1}^{N} b_i y_i)/(A + \sum_{i=1}^{N} a_i/y_i)]^{1/2} \qquad (3.18)$$

Let L = Least common multiple of Y-vector.

$$Q' = (\frac{Q'}{L} \updownarrow) L$$

With the updated value of Q', the X and Y vectors are also updated as,

$$x_i' = Q'/y_i, \quad \forall\ i$$

and

$$y_i' = [\frac{1}{x_i'^{2}}(B/A) + 0.25]^{1/2}\updownarrow, \quad \forall\ i \qquad (3.20)$$

With the available new Y-vector, the lot size Q' and X-vector are modified using (3.18) and (3.19). With every iteration the total cost decreases and the heuristic procedure stops when no further reduction in cost is possible or Q', X, Y vectors stablise. The various steps of the heuristic procedure are summarised below.

Heuristic Algorithm:

Step 1: Calculate the co-efficients of A, B, a_i, b_i. Set $I = 1$, $TC = 38E + 11$ (a high value).

Step 2: Calculate

$$Q = (B/A + 0.25)^{1/2} \updownarrow$$

$$x_i = (b_i/a_i + 0.25)^{1/2} \updownarrow \quad \forall \; i$$

$$y_i = [\frac{1}{x_i^2}(B/A) + 0.25]^{1/2} \updownarrow \; \forall \; i$$

Step 3: L = Least common multiple of Vector Y

$$Q' = [(B + \sum_{i=1}^{N} b_i y_i)/(A + \sum_{i=1}^{N} a_i/y_i) + 0.25]^{1/2}$$

$$Q = (\frac{Q'}{L} \updownarrow)\, L$$

If $(Q' = 0)$ then $Q' = L$

$$x_i' = Q'/y_i, \quad y_i' = [\frac{1}{x_i'^2}(B/A) + 0.25]^{1/2} \updownarrow$$

$$TC = AQ' + B/Q' + \sum_{i=1}^{N} \phi(y_i')$$

Step 4: If $Q' = Q$ and
$x_i' = x_i \ \forall \ i$ and
$y_i' = y_i \ \forall \ i$ or
$TC' > TC$ GO TO STEP 6
otherwise Step 5.

Step 5: Set $I = I + 1$
Update $Q = Q$,
$y_i = y_i' \ \forall \ i$
$x_i = x_i' \ \forall \ i$
$TC = TC'$
GO TO STEP 3.

Step 6: Write Q, x_i, y_i, TC as the heuristic solution. STOP.

3.2.7 Optimisation Procedure:

The heuristic solution gives an upper bound on the cost of the optimal solution. Relaxing the integer constraint on x_i in (3.9), the minimum of $\theta(x_i)$ can be given from Lemma 2 as $2(a_i b_i)^{1/2}$. Then we have, from (3.9)

$$\overline{TC} = AQ + \frac{B}{Q} + 2 \sum_{i=1}^{N} (a_i b_i)^{1/2} \tag{3.21}$$

Substituting the heuristic solution cost as $\overline{TC}$ and simplifying (3.20) we get,

$$Q^2 - 2\alpha Q + \beta = 0 \tag{3.22}$$

where,

$$\alpha = [\overline{TC} - 2\sum_{i=1}^{N} (a_i b_i)^{1/2}]/2A$$

$$\beta = B/A$$

By solving (3.22), an upper and a lower bound on the optimal lot size can be established as,

$$Q_U = \alpha + (\alpha^2 - \beta)^{1/2}$$

$$Q_L = \alpha - (\alpha^2 - \beta)^{1/2}$$

Using Q_L as the starting point, the entire range of lot size is to be scanned for the optimal solution.

For the development of an efficient scanning procedure, let us consider the effect of change in value of some y_i over the given objective function. The part of the objective function influenced by y_i is given by,

$$\phi(y_i) = (\frac{b_i}{Q})\, y_i + (a_i Q)/y_i$$

Let us find the condition under which a change in y_i from y_i to $y_i + 1$ will result in

$$\phi\,(y_i + 1) \leq \phi\,(y_i)$$

Let us assume,

$$\phi\,(y_i + 1) \leq \phi\,(y_i)$$

$$(\frac{b_i}{Q})\,(y_i + 1) + \frac{a_i Q}{y_i + 1} \leq (\frac{b_i}{Q})\, y_i + (a_i Q)/y_i$$

After rearranging we get,

$$\frac{b_i}{Q} \leq \frac{a_i Q}{y_i(y_i+1)}$$

Thus,

$$Q \geq \left[\frac{b_i}{a_i} y_i (y_i + 1)\right]^{1/2} \tag{3.23}$$

From the above inequality, we infer that the starting points of ranges associated with changing y_i to $y_i + 1$ for each stage separately are,

$$Q_i' = \left[\frac{b_i}{a_i} y_i(y_i + 1)\right]^{1/2} \tag{3.24}$$

In the scanning process, starting with Q_L we first, establish the values for Y-vector using (3.16) and (3.17). With the available Y-vector the lot size is obtained using (3.18). Then the new range of lot sizes for change in y_i to y_i+1, for every i are established using (3.23). If the $\min\limits_{\forall i} (Q_i')$ exceeds the upper bound, we have reached an optimal solution, otherwise the y_i corresponding to $\min\limits_{\forall i} (Q_i')$ can be changed to $y_i + 1$ as this decreases the value of $\phi(y_i)$. The entire scanning process is repeated again until all Q_i' s fall outside the upper bound Q_L. The various steps in the optimising algorithm are given below.

<u>Step 1</u>: Set,

$$\overline{TC} = (TC)_{heuristic}$$

$$\overline{Q} = (Q)_{heuristic}$$

$$\overline{x}_i = (x_i)_{heuristic} \quad \forall \; i$$

$$\overline{y}_i = (y_i)_{heuristic} \quad \forall \; i$$

<u>Step 2</u>: $\alpha = [\,\overline{TC} - 2 \sum_{i=1}^{N} (a_i b_i)^{1/2}]/_{2A}$

$$\beta = B/A$$

$$Q_L = \alpha - (\alpha^2 - \beta)^{1/2},$$

$$Q_U = \alpha + (\alpha^2 - \beta)^{1/2}$$

$$x_i' = (b_i/a_i + 0.25)^{1/2} \updownarrow \quad \forall \; i$$

$$y_i' = \left(\frac{Q_L}{x_i'}\right) \updownarrow \quad \forall \; i$$

<u>Step 3</u>: L = Least common multiple of Y-vector

$$Q' = [\{(B + \sum_{i=1}^{N} b_i y_i)/(A + \sum_{i=1}^{N} a_i/b_i)\} + 0.25]^{1/2}$$

$$Q' = \left(\frac{Q'}{L}\right) L$$

<u>Step 4</u>: $x_i' = Q'/y_i' \quad \forall \; i$

$$TC' = AQ' + B/Q' + \sum_{i=1}^{N} \phi(y_i')$$

If ($TC' > \overline{TC}$) GO TO STEP 6.

<u>Step 5</u>: Set, $\overline{TC} = TC'$; $\overline{Q} = Q'$;

$\overline{y}_i = y_i' \;\forall\; i$; $\overline{x}_i = x_i' \;\forall\; i$

<u>Step 6</u>: $Q_i'' = [\frac{b_i}{a_i} y_i' (y_i' + 1)]^{1/2} \;\forall\; i$

$M = \min\limits_{\forall\, i} (Q_i'')$; $j = K$: $Q_K'' = M$

<u>Step 7</u>: If $M > Q_U$ GO TO STEP 8

$y_j' = y_j' + 1$; GO TO STEP 3.

<u>Step 8</u>: Write $\overline{TC}$, $\overline{Q}$, $\overline{y}_i$, $\overline{x}_i$ as the optimum results. STOP.

3.2.8 <u>Efficiency of the Scanning Procedure</u>:

The efficiency of the scanning procedure can be measured in terms of the number of iterations involved as compared to the complete enumeration method.

Let the feasible value of y_i falls between y_i^{min} and y_i^{max}. Let K_i be the total number of feasible values of y_i. For a N-stage problem the number of iterations for complete enumeration would be

$$I_e = K_1 K_2 \ldots K_N$$

In the scanning procedure used in the optimising algorithm only one y_i is changed to y_i+1 in every iteration and this can happen K_i-1 times for each i. Thus for a N-stage problem, the number of iterations would be

$$I_s = (K_1-1) + (K_2-1) + \ldots + (K_N - 1) + 1$$

$$= \sum_{i=1}^{N} K_i - N + 1$$

For $N = 4$; $K_i = 10 \quad \forall i = 1,4$

$$I_e = 10^4; \; I_s = 37.$$

Thus there is considerable reduction in number of iterations in the scanning process.

3.2.9 Numerical Example:

A sample problem is solved numerically illustrating the various steps in the algorithm. The input data for the problem is given in Table 3.4.

Table 3.4: Input data for the numerical example.

$D = 5000$ per year; Total hours available in a year = 2880.

Stage	m_i	F_i	h_i	T_i
1	12.0	18.0	0.40	0.50
2	10.0	15.0	0.50	0.75
3	8.0	16.0	0.60	0.60

Heuristic Solution:

Step 1: $A = 0.2566$; $B = 245000$;

$a_i = \{0.1157 \;\; 0.117 \;\; 0.1388\}$; $b_i = \{2500 \;\; 3750 \;\; 3000\}$

$TC = 38E + 11$ (a high value)

Step 2: $Q = 977$; $x_i' = \{147, 180, 147\}$; $y_i' = \{7, 5, 7\}$

Step 3: $L = 35$; $Q' = 978$; $Q' = 980$; $x_i' = \{140, 196, 140\}$

$TC' = 618.1966$

Step 4: $618.1966 < 38E + 11$: GO TO STEP 5.

Step 5: $I = 2$; $Q = 980$; $y_i = \{7\ 5\ 7\}$; $x_i = \{140\ 196\ 140\}$

$TC = 618.1966$; GO TO STEP 3.

Step 3: $L = 35$; $Q' = 978$; $Q' = 980$; $x_i' = \{140, 196, 140\}$

$y_i' = \{7, 5, 7\}$; $TC' = 618.1966$

Step 4: As $Q' = Q$; $x_i' = x_i$; $y_i' = y_i$ and $TC' = TC$ the procedure stops. Write

$$(Q)_{\text{heuristic}} = 980$$
$$(x_i)_{\text{heuristic}} = \{140, 196, 140\}$$
$$(y_i)_{\text{heuristic}} = \{7, 5, 7\}$$
$$(TC)_{\text{heuristic}} = 618.1966$$

Optimal Solution:

Step 1: $\overline{TC} = 618.1966$, $\bar{Q} = 980$, $\bar{y}_i = \{7, 5, 7\}$,

$\bar{x}_i = \{140, 196, 140\}$

Step 2: $\alpha = 977.61218$, $\beta = 954793.45$

$Q_L = 947.08$, $Q_U = 1008.1429$

$x_i = \{147\ 180\ 147\}$, $y_i = \{6\ 5\ 6\}$

Step 3: $L = 30$, $Q = 960$

Step 4: $x_i' = \{160\ 195\ 160\}$, $TC' = 618.3785$

$TC' > \overline{TC}$, GO TO STEP 6

Step 6: $Q_i'' = \{952.638, 986.07, 952.7754\}$, $M = 952.638$, $j = 1$

Step 7: $952.638 < 1008.1429$, $y_1 = 6 + 1 = 7$, GO TO STEP 3

Step 3: $y_i = \{7\ 5\ 6\}$, $L = 210$, $Q' = 974$, $Q' = 1050$

Step 4: $x_i' = \{150\ 210\ 175\}$, $TC' = 616.317$,

$TC' < \overline{TC}$, GO TO STEP 5.

Step 5: $\overline{TC} = 616.317$, $\overline{Q} = 1050$, $\overline{x}_i = \{150, 210, 175\}$,

$\overline{y}_i = \{7, 5, 6\}$

Step 6: $Q_i'' = \{1100.01, 986.07, 952.7754\}$, $j = 3$,

$y_3 = 6 + 1 = 7$, GO TO STEP 3.

Step 3: $L = 35$, $Q' = 978$, $Q' = 980$, $TC' = 618.196$,

$TC' > \overline{TC}$, GO TO STEP 6.

Step 6: $Q_i'' = \{1100.01, 986.07, 1100.17\}$, $j = 2$,

$y_2 = 5+1 = 6$, GO TO STEP 3.

Step 3: $Q' = 990$, $Q' = 1008$.

Step 4: $x_i' = \{144, 168, 144\}$, $TC' = 618.269$,

$TC' > \overline{TC}$, GO TO STEP 6.

Step 6: $Q_i'' = \{1100.01, 1166.7387, 1100.17\}$, $M = 1100.01$,

$M > Q_L$, GO TO STEP 8.

Step 8: Write,

$(Q)_{opt} = 1050$,

$(x_i)_{opt} = \{150, 210, 175\}$,

$(y_i)_{opt} = \{7, 5, 6\}$

$(TC)_{opt} = 616.317$

3.2.10 Computational Experience:

The algorithm has been coded in Fortran-10 and implemented on DEC-1090 system. Number of problems of varied sizes (Number of stages) were tested. The input parameters viz., demand, setup costs, inventory holding costs and transportation cost, were selected randomly. The ranges selected for the various input parameters are given in Table 3.5. For each

Table 3.5: Ranges of input data.

No. of stages	D	m_i	F_i	h_i	T_i
1-10	1000-15000	5-40	10-40	0.10-0.80	0.25-3.0

problem size ten problems were solved. It was observed that in most of the cases the heuristic solution was obtained in less than three iterations and in no case it exceeded five iterations. In all the cases except two cases, the heuristic solution was found to be optimal. One of the cases in which the heuristic solution was not found to be optimal is given as an illustrative example in the previous section.

Though the heuristic procedure gave the optimal solution in most of the cases, the optimality could not be guaranteed. Further, it was found that of the total CPU time for solving a given size problem, the heuristic procedure consumed less than 50 percent of the time. It optimality is not be guaranteed, considerable saving in the computational effort may result in by simply using the heuristic procedure to obtain the solution of the problem.

The effect of number of stages on the computational time was investigated and is presented in Table 3.6. For single and two stage problems the computational time was found

Table 3.6: Computational performance.

No. of stages	Average CPU time in milli sec.
1	92.5
2	75.0
3	34.2
4	36.5
5	40.1
6	53.25
7	74.8
8	154.4
9	226.5
10	645.25

to be more than that of a 3-stage problem. This can be explained because in the single and two stage problems, the number of combinations of X and Y vectors to be evaluated is greater than those for the 3-stage problem. However, for problem of size greater than 3-stages, the computational time is found to increase with number of stages. This is due to higher number of enumerations to be carried out to encompass the total number of stages in the problem. This is shown graphically in Fig. 3.2.

MODEL 3

3.3 VARIABLE LOT SIZE MODEL:

3.3.1 Statement of the Problem:

Consider a GT production cell comprising of multi-stages where in the lot size at each stage is an integer

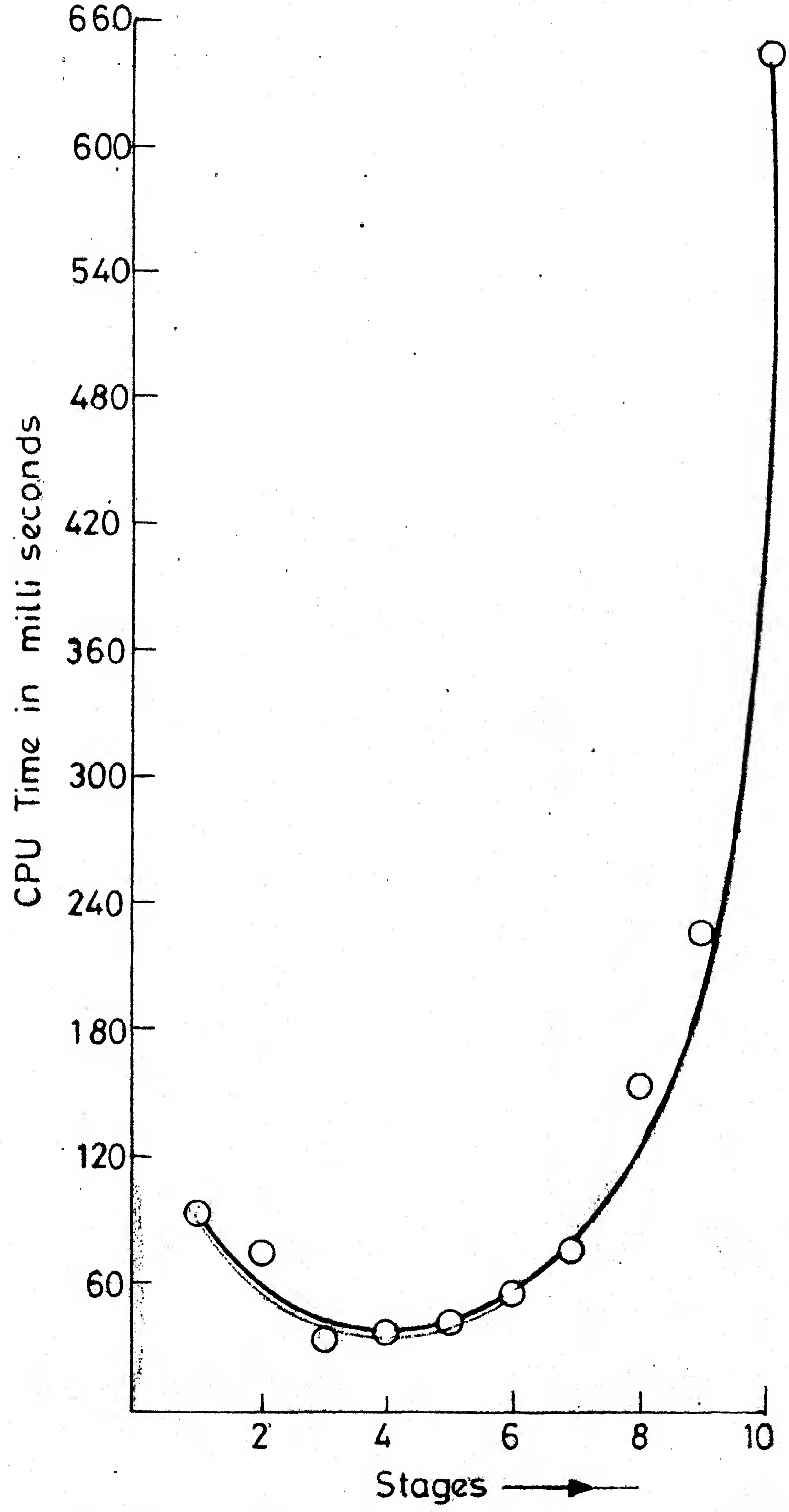


FIG. 3·2 COMPUTATIONAL PERFORMANCE OF CONSTANT LOTSIZE MODEL WITH LOT SPLITTING multiple of the lot size at its succeeding stage. The demand rate is constant over the planning horizon. The problem is to determine the optimal lot size for each stage, so as to minimise the total cost of the production/inventory system.

3.3.2 Assumptions:

1. The production rate at each stage is greater than the demand rate.
2. The demand rate is constant over planning horizon.
3. The setup costs are fixed at each stage.
4. The lot is transported to the next stage only when the entire lot is processed.
5. No shortages are allowed.

3.3.3 Notation:

D	:	Demand rate of the component of the part family
N	:	Number of stages

At any stage i

Q_i	:	Lot size
F_i	:	Fixed setup costs
h_i	:	Inventory holding cost per unit
m_i	:	Machining time
T_i	:	Transportation cost per lot
K_i	:	Positive integer ≥ 1

$F(Q_N, K_i)$: Cost function with lot size $(Q_i = Q_N K_i)$

$t_N(K_i, Q_N)$: Transfer function of K_i and Q_N.

3.3.4 Model Formulation:

For a multi-stage production system the inventory at a stage i is defined as the number of units which have passed through the stage i but not left the system. For such a system Crowston et al (19) have shown that the lot size at each stage should be an integer multiple of the lot size at its succeeding stage. This is known as integrality theorem. The theorem is stated below, without proof.

Integrality Theorem: If i denotes any stage, a(i) denotes its successor stage and N denotes the final stage then there exists a set of optimal lot sizes $\{Q_1, Q_2, \ldots, Q_N\}$ such that for all $i < N$ the ratios,

$$K_i = \frac{Q_i}{Q_{a(i)}} \quad \text{are positive integers.}$$

The proof is given in Crowston et al (19).

The cost function $f(Q_N, K_i)$ at each stage consists of setup costs, inventory holding costs and transportation cost. For each cycle, the setup and transportation cost for stage i will be $(F_i + T_i)$. The total setup and transportation cost of the stage for fulfilling the demand D in lots of Q_i will be $\frac{D}{Q_i}(F_i + T_i)$.

Fig. 3.3 represents the inventory buildup at stage i. The shaded area represents the time weighted inventory at stage i. The shaded area is given by

$$\Delta = \frac{1}{2} Q_i \left(\frac{Q_i}{D} - Q_i m_i \right)$$

The average inventory holding cost is obtained by multiplying the time weighted inventory with number of cycles and the unit inventory holding cost. Average inventory carrying cost is given by the expression,

$$h_i \frac{D}{Q_i} \Delta$$

$$= \frac{1}{2} h_i \frac{D}{Q_i} . Q_i . Q_i \left(\frac{1}{D} - m_i\right)$$

$$= \frac{1}{2} h_i \; Q_i \; (1 - Dm_i)$$

The total cost function for stage i is given by,

$$f(Q_N, K_i) = \frac{D(F_i + T_i)}{Q_i} + \frac{1}{2} h_i \; Q_i \; (1 - Dm_i)$$

$$= \frac{D(F_i + T_i)}{K_i \; Q_N} + \frac{1}{2} h_i \; K_i \; Q_N \; (1 - Dm_i)$$

The objective is to minimize the total cost for all stages. This can be written as,

$$TC = \text{Min} \{ f(Q_N, K_1) + f(Q_N, K_2) + \ldots + f(Q_N \; K_N) \}$$

$$\text{s/t} \quad Q_i = K_i Q_N \quad \text{for } i = 1, 2, \ldots, N-1 \qquad (3.25)$$

$$K_N = 1.$$

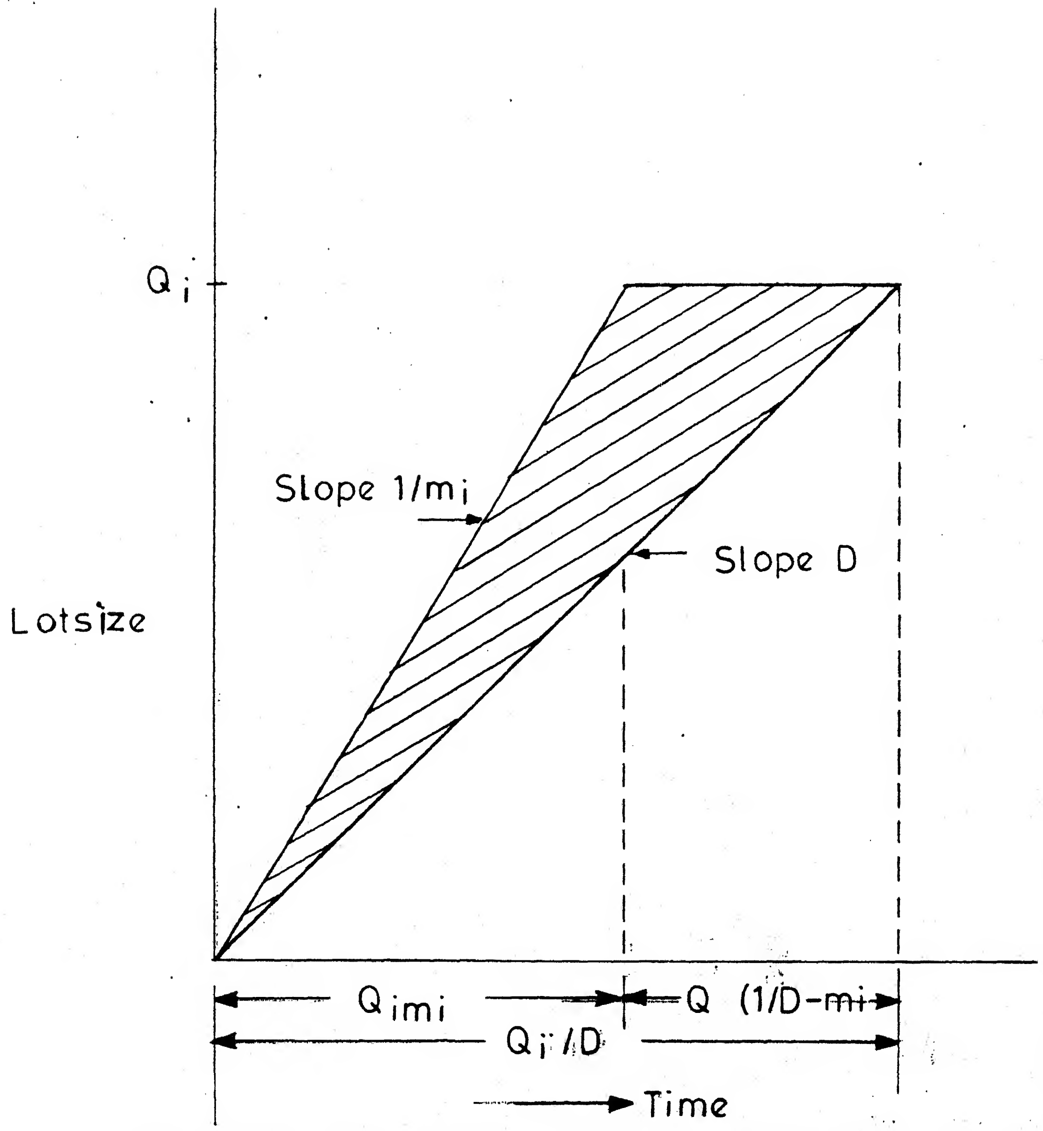


FIG. 3·3 INVENTORY HOLDING COST FUNCTION AT STAGE i

3.3.5 Solution Methodology:

The problem given by (3.25) can be solved using Dynamic Programming (DP). Dynamic programming can be used only if the cost function is decomposable. Mathematically, the total cost function is decomposable if it satisfies the following theorem.

Decomposition Theorem: If a real-valued return function $\phi_N(f_1, f_2, \ldots, f_N)$ satisfies

a) separability condition, i.e.,

$$\phi_N (f_1, f_2, \ldots, f_N) = \phi_{N-1} + f_N$$

where ϕ_{N-1} is a real valued function.

b) ϕ_N is monotonic ~~non-decreasing~~ function of ϕ_{N-1} for every f_N, then ϕ_N is said to be decomposable.

Since the total cost function given in (3.25) is separable and monotonically non-decreasing, it is decomposable.

The dynamic programming model formulated can be represented diagramatically as shown in Fig. 3.4. The recursive relations can be written as,

$$K_{i-1} = t_n (K_i, Q_N) \tag{3.26}$$

$$TC_N = \underset{\text{all predecessors}}{\text{Min}} (TC_{N-1}) + f(Q_N, K_N) \tag{3.27}$$

$$TC_{N-1} = \underset{\text{all predecessors}}{\text{Min}} (TC_{N-2}) + f(Q_N, K_{N-1}) \tag{3.28}$$

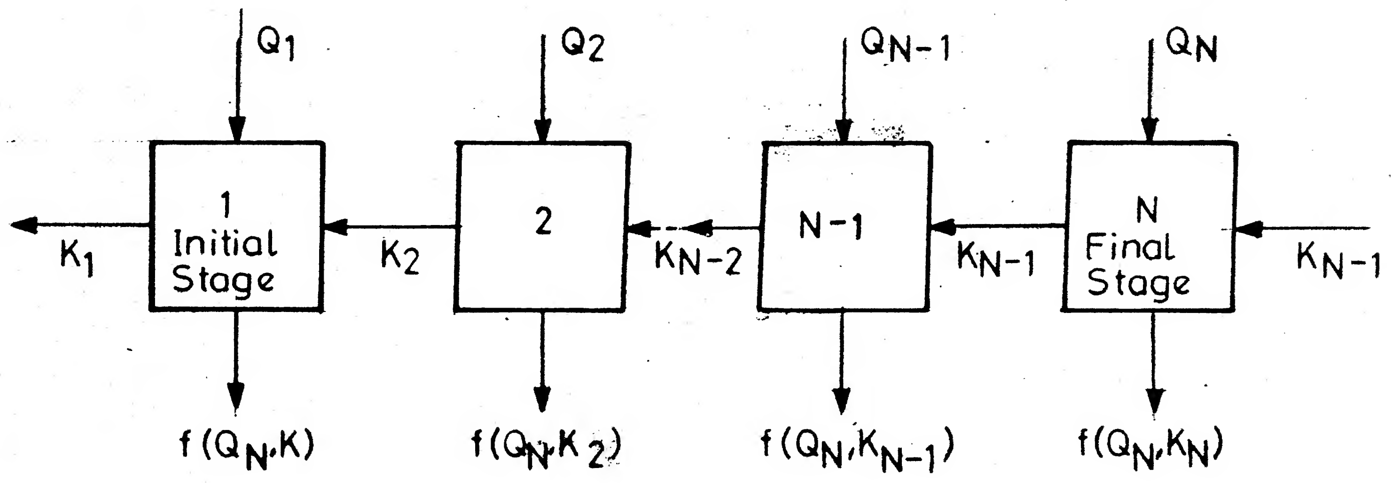


FIG. 3·4 DP SOLUTION METHODOLOGY BY FORWARD RECURSION

From these recurrence relations, the total cost function can be optimised, using forward recursion. It involves optimising the final stage given the initial condition that $K_N = 1$. Then the preceeding stage is optimised for K_{N-1} and Q_N. For the available K-vector an optimal lot size Q_N is found. This continues till the final stage is reached.

3.3.6 Computational Experience:

The dynamic programming algorithm has been coded in Fortran -10 and implemented on DEC-10 system. Number of problems of varied sizes (number of stages) were tested. The input parameters viz., demand, setup costs, inventory holding costs and transportation cost were selected randomly. The ranges selected for the various input parameters are given in Table 3.7.

Table 3.7: Ranges of input data.

No. of stages	D	m_i	F_i	h_i	T_i
1 - 10	1000-20000	5-40	10-40	0.10-0.80	0.25-3.0

The average computational time required for solving problems of varied sizes (in terms of number of stages) was investigated. For each problem size, five problems were solved. Table 3.8 gives the average computational time requirements which are presented graphically in Fig. 3.5. It is observed that the computational time requirements vary ~~exponentially~~ Polynomially [illegible] with the number of stages.

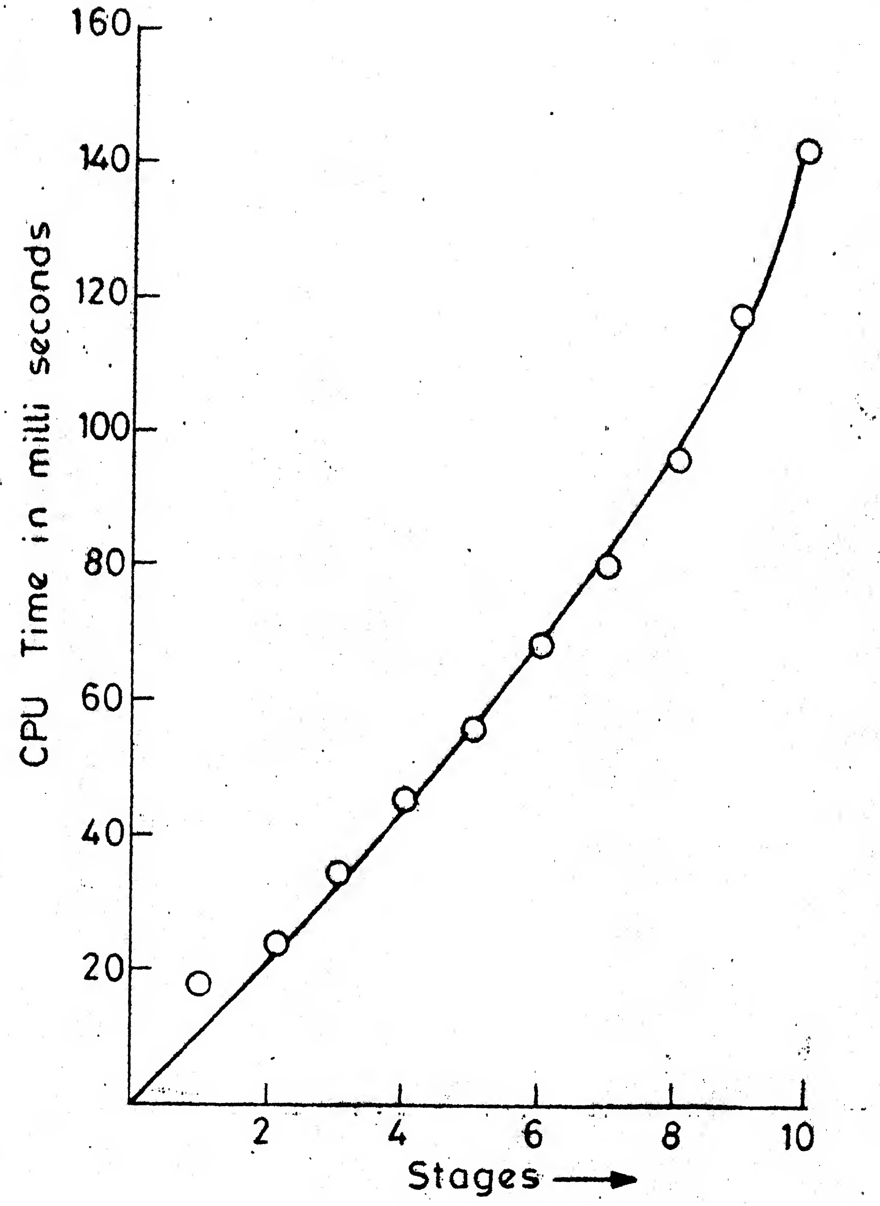


FIG. 3·5 COMPUTATIONAL PERFORMANCE OF DP ALGORITHM

Table 3.8: Average computational time vs. number of stages.

No. of stages	Average computational time in milli sec.
1	18.9
2	29.1
3	34.2
4	45.0
5	54.1
6	66.4
7	80.3
8	96.5
9	119.1
10	144.6

MODEL 4:

3.4 VARIABLE LOT SIZE MODEL WITH LOT SPLITTING:

3.4.1 Statement of the Problem:

Consider a GT production cell where in the lot size at each stage should be an integer multiple of the lot size at its succeeding stage. In addition, the production lot at each stage can be split into batches. A batch can be transported to the next stage even when the lot to which the batch belongs is still being processed at the current stage. The lot size, batch size and number of batches at each stage should be integers. The problem is to find the variable lot sizes, the number of batches and the batch size at each stage such that the total cost of the production-inventory system for the cell is minimised.

3.4.2 Assumptions:

1. Demand rate is constant over time horizon.
2. Inventory carrying costs are linear in nature.
3. Inter-transfer time is negligible.
4. The lot size at stage i is restricted by the capacity of the stage.
5. The maximum batch size depends upon the load carrying capacity of transport equipment at that stage.

3.4.3 Notation:

For a given component of a part family,

D	:	Demand rate,
N	:	Number of stages,

At any stage i,

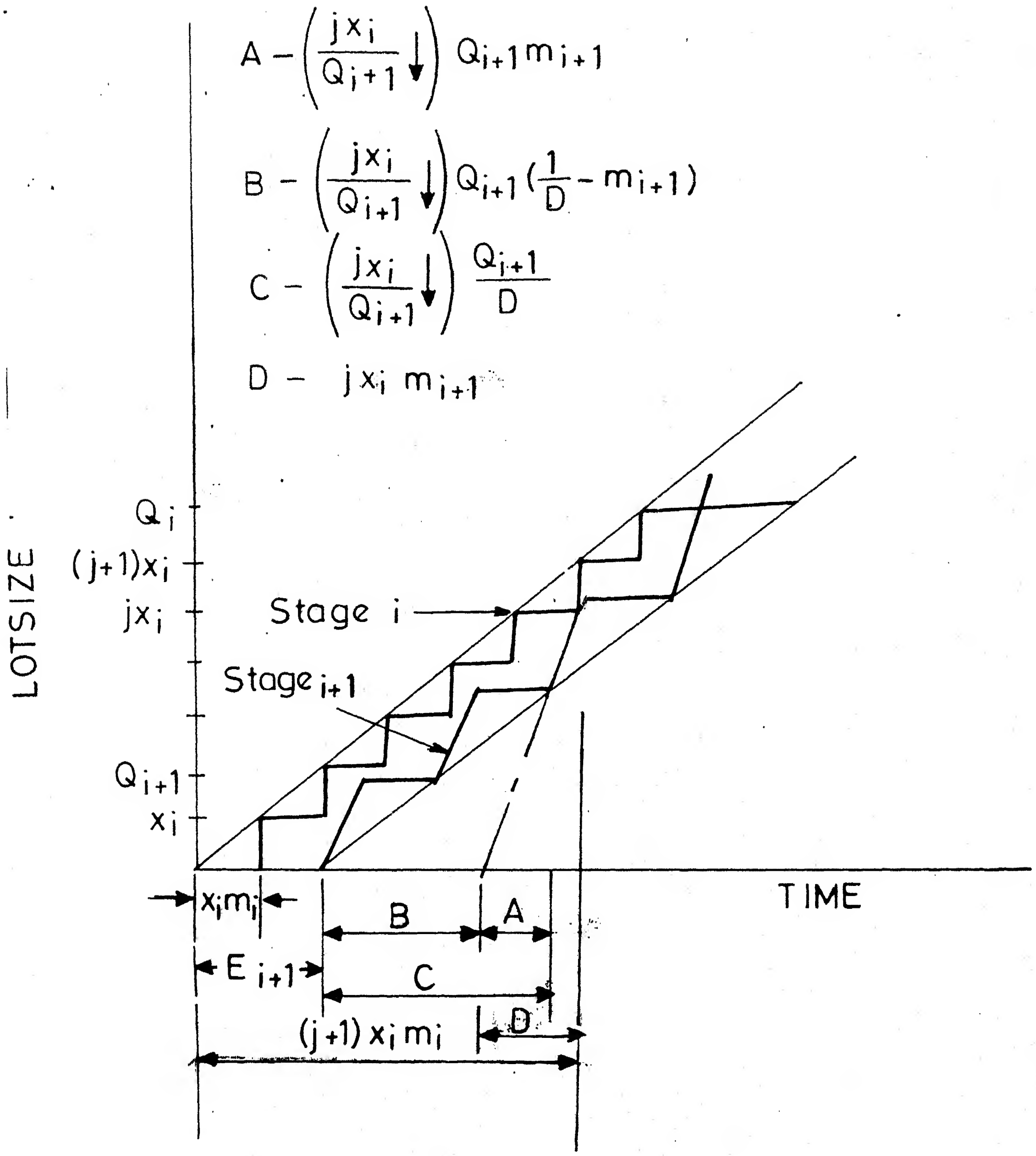
Q_i	:	Lot size produced,
m_i	:	Machining time,
h_i	:	Inventory carrying cost per unit,
x_i	:	Batch size,
y_i	:	Number of batches,
g_i	:	Maximum load carrying capacity,
L_i	:	Maximum lot size permitted,
S_i	:	Integer ratio, ($S_i = Q_{i-1}/Q_i$)
E_{i+1}	:	Earliest time at which production can be started at stage i+1.

$R\uparrow$:	Real value R rounded to higher integer,
$R\downarrow$:	Real value R rounded to lower integer,
$R\updownarrow$:	Real value R rounded to nearest integer.

3.4.4 Model Formulation:

Fig. 3.6 shows the inventory buildup at stages i and i+1, when the machining time at stage i is greater than at stage i+1. The upper slanted line represents the uninterrupted production at stage i with a slope of $1/m_i$. From this stage y_i number of equal sized batches ($x_i = Q_i/y_i$) are transported to the next stage i+1 as and when they are completed at stage i.

Since $m_i > m_{i+1}$, production at stage i+1 cannot be started as soon as the first batch arrives at stage i+1. There should be some elapsed time after which only production at stage i+1 can be started. The second step function represents production at stage i+1. However, the second slanted line which appears below the first one, should satisfy the continuous cumulative demand. The minimum earliest start time for stage i+1 would depend on where the two step functions meet. This is necessary to satisfy the condition that there can not be production of the component at stage i+1 without its being processed at the earlier stage i. Using this condition, the total elapsed time for both the stages can be obtained. Let j, $j = \{1,2,\ldots, y_i\}$ represent the batch at stage i which is



A – (jx_i / Q_{i+1} ↓) Q_{i+1} m_{i+1}
B – (jx_i / Q_{i+1} ↓) Q_{i+1} (1/D – m_{i+1})
C – (jx_i / Q_{i+1} ↓) Q_{i+1} / D
D – jx_i m_{i+1}
LOTSIZE
Q_i
(j+1)x_i
jx_i
Q_{i+1}
x_i
Stage i
Stage i+1
TIME
x_i m_i
B
A
E i+1
C
D
(j+1) x_i m_i

FIG. 3·6 TIME-WEIGHTED INVENTORY BETWEEN STAGES i AND i+1 WHEN $m_i > m_{i+1}$

currently being processed at stage i+1. The total elapsed time at stage i is given by,

$$ET_i = (j+1)\, x_i m_i \tag{3.29}$$

The total elapsed time at stage i+1 can be expressed as,

$$ET_{i+1} = E_{i+1} + \frac{jx_i}{Q_{i+1}} \downarrow \left(\frac{Q_{i+1}}{D}\right) - \frac{jx_i}{Q_{i+1}} \downarrow \left(Q_{i+1}\, m_{i+1}\right) + jx_i m_{i+1} \tag{3.30}$$

Equating (3.29) and (3.30) and simplifying, the earliest start time at stage i+1 is given by,

$$E_{i+1} = (j+1)\, x_i m_i - jx_i m_{i+1} - \left(\frac{jx_i}{Q_{i+1}} \downarrow\right) Q_{i+1} \left(\frac{1}{D} - m_{i+1}\right)$$

The condition that there can be production at stage i+1 only when stage i supplies it implies that the

elapsed time at i+1 $\geq$ elapsed time at i

$$\Longrightarrow E_{i+1} + jx_i\, m_{i+1} + \frac{jx_i}{Q_{i+1}} \downarrow Q_{i+1} \left(\frac{1}{D} - m_{i+1}\right) \geq (j+1)\, x_i m_i$$

$$\text{for } 0 \leq j \leq y_i - 1$$

Rearranging and simplifying we get,

$$E_{i+1} = x_i m_i + \max_{0 \leq j \leq y_i - 1} [\phi(j)] \tag{3.31}$$

where,

$$\phi(j) = jx_i\, (m_i - m_{i+1}) - \left(\frac{jx_i}{Q_{i+1}} \downarrow\right) Q_{i+1} \left(\frac{1}{D} - m_{i+1}\right)$$

For the case when $m_i \leq m_{i+1}$, the production at stage i+1 can be started as soon as first batch comes out from stage i.

This is because machining time at stage i is less compared to that at stage i+1 and so there will be enough units at stage i+1 for continuous production.

Hence the maximum earliest start time would be,

$$E_{i+1} = x_i m_i$$

Since $(m_i - m_{i+1})$ is always non positive $\phi(j)$ equals zero hence the expression (3.31) can be used to determine E_{i+1}.

To find the average inventory holding cost, let us first determine the time-weighted inventory. This is given by the shaded area in the Fig. 3.7. The area of the shaded portion is found by subtracting the areas of triangles formed from the trapezoid.

Area of the trapezoid is given by,

$$\Delta = [E_{i+1} + \{Q_i/D_i - (Q_i m_i - E_{i+1})\}] \frac{Q_i}{2}$$

Area of the triangles can be written as,

$$\Delta\Delta = \frac{1}{2} Q_{i+1} \quad Q_{i+1} \left(\frac{1}{D} - m_i \right) \frac{Q_i}{Q_{i+1}}$$

Therefore, the shaded area can be expressed by,

$$\frac{1}{2} Q_i \left[2E_{i+1} + Q_i \left(\frac{1}{D} - m_i \right) - Q_{i+1} \left(\frac{1}{D} - m_{i+1} \right)\right]$$

Multiplying the time-weighted inventory by the inventory holding costs h_i and number of cycles, we get the average inventory holding cost expression as,

$$\frac{1}{2} Q_i \left[2E_{i+1} + Q_i \left(\frac{1}{D} - m_i \right) - Q_{i+1} \left(\frac{1}{D} - m_{i+1} \right) h_i \frac{D}{Q_i}\right.$$

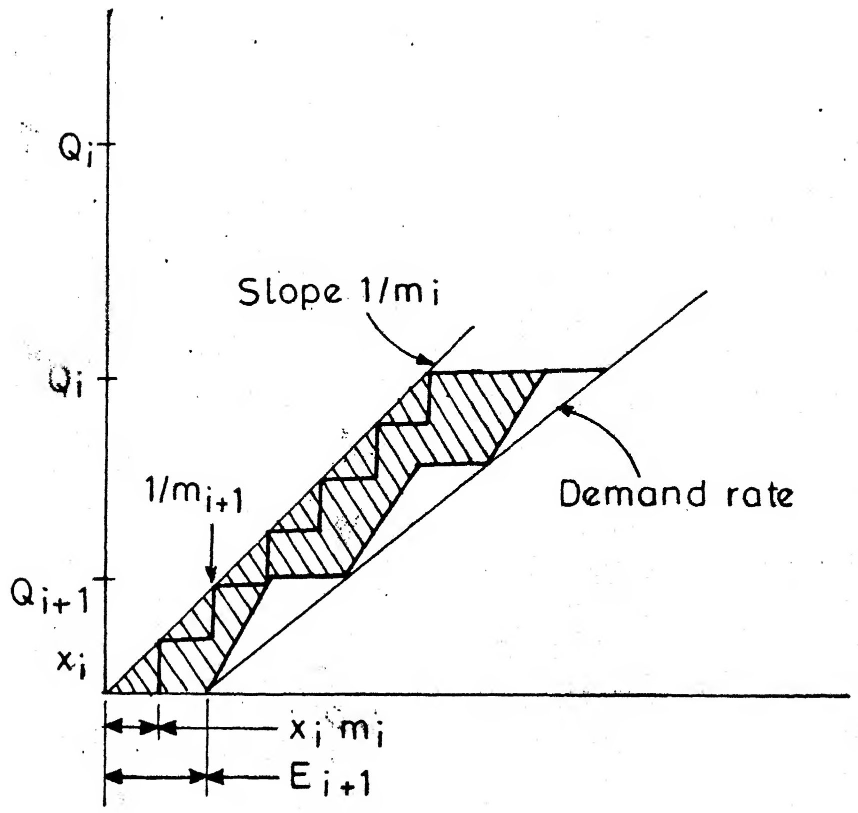


FIG. 3·7 INVENTORY BUILDUP BETWEEN STAGES i AND $i+1$ WHEN $m_i > m_{i+1}$

By adding the total setup costs and total transportation costs, the total cost expression can be written as,

$$TC = D \sum_{i=1}^{N} \left[\left(\frac{F_i + b_i T_i}{Q_i} \right) + \frac{1}{2} h_i \left[2E_{i+1} + Q_i \left(\frac{1}{D} - m_i \right) - Q_{i+1} \left(\frac{1}{D} - m_{i+1} \right) \right\} \right] \qquad (3.32)$$

We have the following constraints to be satisfied at stage i,

a) The batch size at stage i cannot exceed the maximum load carrying capacity at stage i,

$$x_i \leq g_i \qquad \text{for all } i = 1, \ldots, N$$

b) The lot sizes follow the Crowston integrality theorem (19),

$$S_i = \frac{Q_{i-1}}{Q_i} \qquad \text{for all } i = 2, \ldots, N$$

c) The lot size at stage i cannot exceed the maximum lot size permitted at stage i,

$$Q_i \leq L_i \qquad \text{for all } i = 1, \ldots, N$$

d) The lot size is to be divided into equal sized batches,

$$y_i = Q_i/x_i, \ \text{integer} \qquad \text{for all } i = 1, \ldots, N$$

Using $h_o = 0$, the complete optimisation problem can be written in terms of Q_i as given below:

$$\text{Minimise } TC = D \sum_{i=1}^{N} \left[\frac{F_i}{Q_i} + Q_i \left(\frac{1}{D} - m_i \right) (h_i - h_{i-1})/2 + h_i E_{i+1} + \frac{T_i}{x_i} \right] \qquad (3.33)$$

Hence for the above value of j, E_{i+1} would be minimum. This gives the lower bound on E_{i+1}. Substituting $j = \frac{Q_{i+1}}{x_i}\uparrow - 1$ in (3.31) we get,

$$\bar{E}_{i+1} = x_i m_i + \phi \left(\frac{Q_{i+1}}{x_i}\uparrow - 1\right)$$

Lemma 3: For a real value R, we have,

$$\left(\frac{R\uparrow - 1}{R\uparrow}\right)\downarrow = 0$$

Proof: Always $\frac{R\uparrow - 1}{R\uparrow}$ would be less than unity. Rounding to the lower integer always results in zero. Using the above lemma, and simplifying the expression for $\bar{E}_{i+1}$, we get,

$$\bar{E}_{i+1} = x_i m_i + \left[\left(\frac{Q_{i+1}}{x_i}\uparrow - 1\right) x_i (m_i - m_{i+1})\right] \qquad (3.38)$$

The two lower bounds for both cases can be combined into one by introducing a variable α_i as follows:

$$\bar{E}_{i+1} = x_i \left[m_i + (\alpha_i - 1)(m_i - m_{i+1})\right] \qquad (3.39)$$

where,

$$\alpha_i = \begin{cases} 1 & \text{if } m_i < m_{i+1} \\ \frac{Q_{i+1}}{x_i}\uparrow & \text{if } m_i \geq m_{i+1} \end{cases}$$

3.4.5.2 Bounds on Total Cost: To find a lower bound on total cost, the integrality restrictions on S_i, α_i, y_i are relaxed and lower bounds on the earliest start time are substituted for E_{i+1}.

Relaxing integer restriction on α_i, the expression for $\bar{E}_{i+1}$ is given by,

$$\bar{E}_{i+1} = (1-\delta_i)\, x_i m_i + \delta_i\, (x_i m_{i+1} + Q_{i+1}\, (m_i - m_{i+1})] \tag{3.40}$$

where,

$$\delta_i = \begin{cases} 0 & \text{if } m_i < m_{i+1} \\ 1 & \text{if } m_i \geq m_{i+1} \end{cases}$$

Substituting (3.40) in (3.31) and writing all the terms in terms of Q_i we get lower bound on total cost $\bar{TC}$

$$\bar{TC} = D \sum_{i=1}^{N} \Big[\frac{F_i}{Q_i} + Q_i \,\{(\frac{1}{D} - m_i)(h_i - h_{i-1})/2 + \delta_{i-1}\, h_{i-1}\, (m_{i-1} - m_i)\} + x_i h_i\, \{(1-\delta_i) m_i + \delta_i m_{i+1}\} + \frac{T_i}{x_i} \Big]$$

This can be written as,

$$\bar{TC} = D \sum_{i=1}^{N} \Big[\frac{A_i}{Q_i} + B_i Q_i + H_i x_i + \frac{G_i}{x_i} \Big]$$

where

$$A_i = F_i$$

$$B_i = [(\frac{1}{D} - m_i)(h_i - h_{i-1})/2 + \delta_{i-1}\, h_{i-1}\, (m_{i-1} - m_i)]$$

$$H_i = h_i [(i-\delta_i) m_i + \delta_i\, m_{i+1}]$$

$$G_i = T_i$$

Hence the relaxed version of the original problem can be written as,

Minimise $\overline{TC}$

$$\begin{aligned} \text{s.t.} \quad & Q_i \leq L_i && \forall\, i = 1,\ldots, N \\ & x_i \leq g_i && \forall\, i = 1,\ldots, N \\ & Q_{i+1} \leq Q_i && \forall\, i = 1,\ldots, N \\ & x_i \leq Q_i && \forall\, i = 1,\ldots, N \\ & Q_i,\ x_i > 0 \end{aligned} \tag{3.41}$$

The above problem can be solved to find a lower bound on the cost of the original problem. Using this feasible solution close to the lower bound is established by the heuristic procedure described in the next section.

3.4.6 Heuristic Procedure:

3.4.6.1 Outline of the procedure:

The relaxed version of the problem solved in the previous section gives a lower bound on cost and a set of Q_i s. With this the best integer ratios S_i s are established. With the given S_i values the new lot size at all stages are modified. The number of batches at each stage, are established using Q_i, S_i values, by an efficient search method.

With a complete solution available i.e. S, Q, Y vectors, the lot sizes are modified using the eq. (3.32). The entire procedure is repeated until no further reduction in cost is possible. The algorithm converge as the total cost function is a non-decreasing function and there are finite combinations of S and Y vectors.

3.4.6.2 Development of the Procedure:

Initially, with the available solution for the relaxed problem, the S_i values can be established by rounding off procedure,

$$S_K = \left[\frac{Q_{K-1}}{Q_N(S_{K+1}\ S_{K+2}\ \ldots\ S_{N+1})} \right] \updownarrow \tag{3.42}$$

for $K = N, N-1, \ldots, 3, 2$

given $S_{N+1} = 1$

The maximum allowable Q_N due to the lot size restriction at each stage is given by,

$$Q_{UN} = \min_{j \leq i \leq N} \left[L_i/(S_N\ S_{N-1} \ldots S_{i+1}) \right] \tag{3.43}$$

So the permissible lot size at Final stage,

$$Q_N = \min\ (Q_{UN},\ Q_N)$$

Given Q_N and S-vector the new range of Q_i's are given by,

$$Q_{K-1} = Q_N\ (S_K\ S_{K+1}\ \ldots\ S_N\ S_{N+1}) \quad \text{for } K = 2,3,\ldots, N-1,N \tag{3.44}$$

To find the Y-vector consider the following cases:

a) When $E_{i+1} = \dot{E}_{i+1}$

In the total cost eq. given by (3.32), convert each x_i into Q_i/y_i and by writing the total cost in terms of y_i, we get,

$$\Theta(y_i) = \left(\frac{T_i}{Q_i}\right) y_i + h_i \frac{Q_i}{y_i} \left[(1-\delta_i)m_i + \delta_i m_{i+1}\right] \tag{3.45}$$

Case (i) When $m_i \leq m_{i+1}$ $\delta_i = 0$

$$\theta(y_i) = \left(\frac{T_i}{Q_i}\right) y_i + \frac{h_i Q_i}{y_i} m_i$$

$$\text{s.t.} \quad y_i \geq Q_i/g_i$$

Since $\theta(y_i)$ is convex function using Lemma 1 in section 3.2,

$$y_i = \left[\frac{Q_i^2}{T_i} h_i m_i + 0.25\right]^{1/2} \updownarrow \tag{3.46}$$

To satisfy the constraint, we have,

$$y_i = \max\left(y_i , \frac{Q_i}{g_i} \uparrow\right) \tag{3.47}$$

Case (ii) When $m_i > m_{i+1}$, $\delta_i = 1$

Using the Lemma 1 in section 3.2, we can show on similar lines that

$$y_i = \left[\frac{Q_i^2}{T_i} h_i m_{i+1} + 0.25\right]^{1/2} \updownarrow \tag{3.48}$$

and

$$y_i = \max\left[y_i, \frac{Q_i}{g_i} \uparrow\right] \tag{3.49}$$

b) If $E_{i+1} = \bar{E}_{i+1}$ then the values of y_i are unchanged. Otherwise the $\theta(y_i)$ would be modified from (3.32) as given below:

$$\theta(y_i) = \left(\frac{T_i}{Q_i}\right) y_i + h_i \max_{0 \leq j \leq y_{i-1}} \left[j x_i (m_i - m_{i+1}) - \frac{j x_i}{Q_{i+1}} \downarrow (1/D - m_{i+1}) \right] \tag{3.50}$$

Let y_i^* be the value of y_i when $E_{i+1} = \bar{E}_{i+1}$, we have already noted that

$$E_{i+1} \geq x_i m_i$$

Therefore, from Eq. (3.43) we have,

$$\theta(y_i^*) \geq \left(\frac{T_i}{Q_i}\right) y_i + \frac{h_i Q_i m_i}{y_i} \qquad (3.51)$$

By solving inequality given by (3.51), two roots of y_i can be obtained. Searching between the two roots for minimum $\theta(y_i)$ would give the optimum value of y_i.

Now one complete set of solution viz.,

Q-Vector, S-vector and Y-vector is now available. This can be substituted in Eq. (3.32) to get a better value of Q_N.

Simplifying (3.32), we get,

$$TC = \frac{W}{Q_N} + ZQ_N \qquad (3.52)$$

where,

$$W = D \sum_{i=1}^{N} (F_i + y_i T_i)/(S_N, S_{N-1} \ldots S_{i-1})$$

$$Z = D \sum_{i=1}^{N} [S_N\, S_{N-1} \ldots S_{i-1} \left(\frac{1}{D} - m_i\right) (b_i - b_{i-1})/2$$

$$+ \delta_{i-1}\, b_{i-1}\, (m_{i-1} - n_i) + h_i \left\{\frac{(1-\delta_i)}{y_i} m_i\right.$$

$$\left. + \frac{\delta_i m_{i+1}}{y_i} \right\}]$$

By solving the expression given in (3.45), we get,

$$Q = \left(\frac{W}{Z} + 0.25\right)^{1/2} \qquad (3.53)$$

The iterative process continues till the lot size obtained in (3.53) stabilises.

The steps in the algorithm can be summarised as below:

Step 1: Solve the relaxed constraint problem given in (3.41).

Step 2: Establish the S_i values from the eq.(3.42). Modify the lot sizes according to equations given by (3.43) and (3.44), I = 0.

Step 3: I = I + 1 , if $m_i < m_{i+1}$, calculate y_i-value according to equations (3.46) and (3.47), otherwise, calculate y_i - values according to equations (3.48) and (3.49).

Step 4: If $E_{i+1} = \bar{E}_{i+1}$, Go to Step 3, otherwise calculate the roots of y_i according to inequality (3.51) and search for the optimum y_i between the two roots. GO TO STEP 3.

Step 5: Calculate Q_N' using eq. (3.52).

Step 6: If $Q_N' = Q$, Go to Step 7, otherwise $Q_N = Q_N'$, I = I+1, GO TO STEP 2.

Step 7: Write the heuristic results, stop.

3.4.7 Computational Experience:

The variable lot size model with lot splitting has been coded in Fortran-10 for DEC 1090 system. Number of problems of varied sizes (Number of stages) were tested. The input parameters viz., demand, setup costs, inventory holding costs and transportation costs were selected randomly. The ranges selected for various input parameters are given in Table 3.9. For each problem size five problems were solved. It was observed that the heuristic solution was obtained in less than five iterations and in no case it exceeded 10 iterations.

The effect of number of stages on the computational time was investigated. Table 3.10 gives the average computational time requirements which are presented graphically in Fig. 3.9.

Table 3.9: Ranges of input data

i	D	m_i	F_i	h_i	T_i
1 - 10	1000-60000	2-40	10.0-40.0	0.1-0.8	0.25-3.0

Table 3.10: Computational performance

No.of stages	Average CPU time in milli seconds
1	20.1
2	31.5
3	39.5
4	56.0
5	70.0
6	85.1
7	107.3
8	129.8
9	161.7
10	185.3

3.4.8 Comparison of Model 2 and Model 4:

The performances of Model 4 and Model 2 have been compared for problems of varied sizes. For each problem the

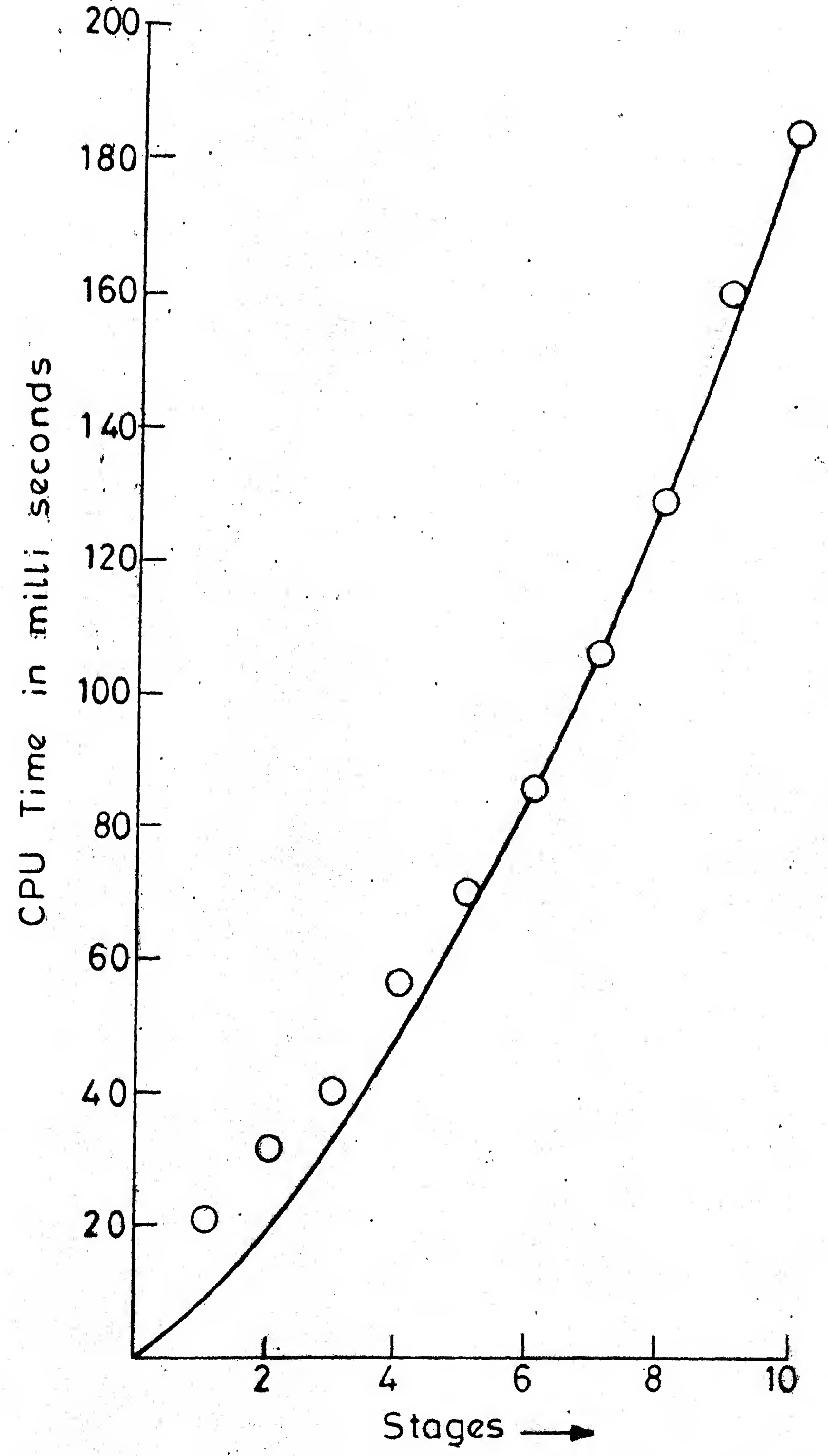


FIG. 3·9 COMPUTATIONAL PERFORMANCE OF VARIABLE LOTSIZE MODEL WITH LOT SPLITTING randomly generated input data was kept the same. The comparison was based on the total cost of operating the production/ inventory system and the computational time. From Table 3.11 we observe that the total cost of the inventory production system is lower for the variable lot size model with lot splitting (Model 4) as compared to the constant lot-size model with lot splitting. However, the results were found to be otherwise for single stage problem. This might have occured due to the heuristic solution procedure followed for model 4. On an average, based on the problems considered, a reduction in total cost of about 10 percent was observed for the Model 4.

Since the amount of reduction would be input data dependent, the only conclusion one can draw is that model 4 should be preferred over model 2. Further, model 4 requires lesser computational time for the same size problem as compared to model 2 as is evident from Tables (3.6) and (3.10).

Table 3.11 Comparison between Models II and IV

Problem Size (No.of Stages)	Total Cost		Percentage Reduction in cost for Model IV
	Model IV	Model II	
1	705.1596	700.029	- 0.7342
2	1311.1646	1409.087	6.949
3	1190.2946	1339.683	11.151
4	1596.0257	1786.867	10.6802
5	1397.6391	1724.614	18.96
6	2086.1632	2184.217	4.489
7	2078.8431	2420.355	14.11
8	2365.1158	2772.907	14.70
9	3426.9020	3781.078	9.367
10	3433.4674	3901.995	12.007

CHAPTER IV

CONCLUSIONS AND SCOPE FOR FURTHER STUDY

4.1 CONCLUSIONS:

In this thesis, we have developed mathematical models and solution methodologies for lot sizing in GT production system. The following conclusions can be made on the models developed and the solution methodologies.

1. The consideration of WIP inventory in GT production system results in lower lot size and lower total cost for the production of a given component of a part family in the cell.
2. The splitting of the lot into batches reduces the WIP inventory significantly.
3. The performance of the heuristic procedure in model 2 is quite encouraging in terms of computational time and its ability to generate optimal solutions.
4. The variable lot size model with lot splitting would be preferable to the constant lot size model with lot splitting.

4.2 SCOPE FOR FURTHER STUDY:

The models presented in this work have been developed based on several assumptions. These assumptions can be relaxed to make the models more realistic. Specific models need to be developed for the following situations.

1. Demand instead of being constant varies with time.
2. Demand is stochastic.
3. Stages comprise of more than one machine and the capacity of the machine at each stage is limited.
4. Setup costs are sequence dependent.

REFERENCES

1. Mitrofanov, Scientific Principles of Group Technology.

2. Burbidge, J.L., Group Technology in Engineering Industry.

3. Jackson, D., Cell System of Production.

4. Arn, C., Group Technology.

5. Edwards, G.A.B., Readings in Group Technology.

6. El-Essawy, I.G.K., Component Flow Analysis - An Approach to Production Systems Design, Production Engineer, V 51 May, 1965.

7. Mc Auley, J., Machine Grouping for Efficient Production, Production Engineer, V. 51, May 1965.

8. Rajagopalan, R., and Batra, J.L., Design of Cellular Production System, A Graph Theoretic Approach, Int. Jr. of Prod. Res. V 11, p. 399.

9. Crookal, J.R. and Baldwin, K.I., An Investigation into Applications of Grouping Principles Using Monte Carlo Simulation, CIRP, 1, 3.

10. Purcheck, G.F.K., A Mathematical Classification as a Basis for the Design of GT Production Cells, Prod. Engr. V 54, p. 55.

11. Mc Cormick, M., Problem Decomposition and Data Recognition by Clustering Technique, Opns. Res. V 20, p. 993.

12. Waghodekar, P.H., and Sahu, S., Machine Component Cell Formation in GT, IJPR, V. 22, p. 937.

13. Petrov, V.A., Flow Line Group Production Planning, Business Publications Ltd., 1968.

14. Hitomi, K., and Ham, I., Group Scheduling Technique for Multi-Product Multi-Stage Manufacturing Industry, Jr. of Engg. For Industry, Aug. 1977, pp. 759.

15. Hitomi, K., Ham, I., Yoshida, T. Optimal Group Scheduling and Machining Speed Decisions Under Due-Date Constraints, Trans. of ASME, Journal of Engg. for Industry, Vol.101, p. 128.

16. Kishore, S., Scheduling of Component Groups in a Multi-Stage Production System, M.Tech. Thesis, IIT Kanpur.

17. Hitomi, H., and Ham, I., Machine Loading Product Mix Analysis, for GT, Jr. of Engg. for Industry, ASME, Paper No. 77-WA/Prod. 21, 1-5.

18. Agrawal, A.K., Approaches for Machine Loading and Product Mix Analysis for Single and Multi-Stage Production Systems Using G.T. Concepts, M.Tech. Thesis, IIT Kanpur.

19. Crowston et.al., Economic Lot Size Determination in Multi-Stage Assembly Systems, Management Science, V. 19, p. 517.

20. Chakravarty, A., Group Technology with Inprocess Inventory Costs, IJPR, V. 19, p. 437.

2L. Ignall and Veinott, Optimality of Myopic Inventory Policies for Several Substitute Products, Mgt. Science, V 15, p. 284.

22. Wagner and Whitin, Dynamic Version of Economic Lot Size Model, Mgt. Science, V. 5, p. 89.

23. Zangwill, W.I., A Back-Logging Model and a Multi-Stage Echelon Model of Dynamic Economic Lot Size Production System - A Network Approach, Mgt. Science, V.15,p.506.

24. Goyal, S.K., Lot Size Scheduling on a Single Machine for Stochastic Demand, Mgt. Science, V. 22, p. 967.

25. Newson, E.F.P., Multi Item Lot Size Scheduling by Heuristic With Fixed Resources and Variable Resources, Mgt. Science, V. 21, p. 1186.

26. Gupta, R.M., and Tompkins, J.A., An Examination of Dynamic Behaviour of Part Families in G.T., IJPR, V. 20, p. 738.

27. Ang, C.L., and Willey, P.C.T., A Comparative Study of the Performance of Pure and Hybrid GT Manufacturing System Using Computer Simulation Techniques, IJPR, V. 22, p. 193.

28. Mittal, K.V., Optimisation Methods in Operations Research and Systems Analysis.

29. Nicholson, T.A.J., Optimisation in Industry.

30. Waghodekar, P.H., and S. Sahu, (1983), Group Technology: A Research Bibliography, OPSEARCH, Vol.20 No.4.

31. Cooper, L., and Cooper, W., Introduction to Dynamic Programming.

32. Sven Dano, Non Linear and Dynamic Programming.

```
      C     ******************************************************************
      C     THIS PROGRAM IS FOR CONSTANT LOT SIZE MODEL WITH LOT SPLITTING.
      C     THE INPUT IS TO BE GIVEN IN THE FOLLOWING FORMAT. DEMAND,
      C     NO. OF STAGES, PRODUCTION RATES, SETUP COSTS, TRANSPORTATION
      C     COSTS & HOLDING COSTS. THE OUTPUT GIVES THE OPTIMUM COST,
      C     OPTIMUM LOT SIZES AND THE NO. OF BATCHES.

      C     [illegible] OF [illegible]STAGES ALLOWED.

      C     ******************************************************************

            INTEGER PR(21),YINIT(20),XINIT(20),XOPT(20)
            DIMENSION YOPT(20),YNEW(20),-,XINIT(20),XNEW(20),YMIN(20)
            DIMENSION AX(20),BX(20),CI(20),PC(20),TRC(20),NUMBER(20)
            INTEGER DINIT,DNEW,DOPT,XMAX(20),UPPER,LOWER,DMIN
            INTEGER LCM,YA(20),TIME1,TIME2,TIME
            LOGICAL MATCH
            COMMON NOS
            READ(23,*)D,NOS
            READ(23,*),(PR(I),PC(I),TRC(I),CI(I),I=1,NOS)
            PCSUM=0,CISUM=0,MATCH=.FALSE.
            PR(NOS+1)=D
            CALL GETTIM(TIME1)
            DO 10 I=1,NOS
            CISUM=CISUM+CI(I)*0.5*ABS(1.0/PR(I)-1.0/PR(I+1))
            PCSUM=PCSUM+PC(I)
            AX(I)=D*CI(I)/AMAX0(PR(I),PR(I+1))
            BX(I)=D*TRC(I)
         10 CONTINUE
            A=D*CISUM;B=D*PCSUM
            WRITE(31,*),A,B,(AX(I),I=1,NOS),(BX(I),I=1,NOS)
            DINIT=INT(SQRT(B/A+0.25)+0.5)
         15 DO 30 I=1,NOS
            XINIT(I)=INT(SQRT(BX(I)/AX(I)+0.25)+0.5)
            XX=XINIT(I)
            YINIT(I)=INT(SQRT(((B/A)/(XX**2))+0.25)+0.5)
         30 CONTINUE
            X=ASUMAG(A,AX,YINIT)
            Y=BSUMAG(B,BX,XINIT)
            DINIT=INT((SQRT(Y/X)+0.25)+0.5)
            CALL LCMCOM(XINIT,NOS,LCM)
            DINIT=LCM*INT((DINIT/LCM+0.5))
            IF([illegible]) DINIT=LCM
```

```
          [illegible]
    200   [illegible]
          [illegible]
          [illegible]
          [illegible] GO TO 140
          [illegible].25)+0.5)
    40    [illegible]
          [illegible]
          [illegible]
          [illegible]
          [illegible]
          [illegible]
          [illegible]
          [illegible]
          [illegible]
          PSUM=0.,QSUM=0.
          DO 120 I=1,NOS
          PSUM=[illegible](I)+PSUM
          QSUM=[illegible](I)+QSUM
    120   CONTINUE
          CNEW=[illegible]+PSUM+QSUM
          IF(CNEW.GT.CX) GO TO 46
          IF([illegible]) MATCH=.TRUE.
          IF(.NOT.(MATCH)) GO TO 100
          DO 45 I=1,NOS
          IF(((MATCH)).AND.(XNEW(I).NE.XINIT(I))) MATCH=.FALSE.
    45    CONTINUE
          IF(.NOT.(MATCH)) GO TO 100
          DO 50 I=1,NOS
          IF((MATCH).AND.(YNEW(I).NE.YINIT(I))) MATCH=.FALSE.
    50    CONTINUE
          IF(.NOT.(MATCH)) GO TO 100
          QMIN=QNEW
          CMIN=CNEW
          DO 55 I=1,NOS
          XMIN(I)=XNEW(I)
          YMIN(I)=YNEW(I)
    55    CONTINUE
          GO TO 332
    46    QMIN=QINIT
          CMIN=CX
          DO 331 I=1,NOS
          XMIN(I)=XINIT(I)
          YMIN(I)=YINIT(I)
```

```
331     [illegible]
332     WRITE(26,99),NOS
99      FORMAT(1X,"NO. OF STAGES: ",I3)
        WRITE(26,61)
61      FORMAT(1X,"HEURISTIC RESULTS")
        WRITE(26,[illegible]),([illegible],[illegible],(XNEW(I),I=1,NOS),(XOPT(I),I=1,NOS))
        [illegible]
        GO TO [illegible]
100     [illegible]
        [illegible]
        DO 98 I=1,NOS
        [illegible]
        [illegible]
98      CONTINUE
        [illegible]
        [illegible]
        GO TO [illegible]
110     TYPE 62
62      FORMAT(2X,"PROBLEM IN CALCULATING XNEW")
300     [illegible]
        SUM=0.0
        DO 110 I=1,NOS
        SUM=SUM+SQRT(AX(I)*BX(I))
110     CONTINUE
        ALPHA=(CTIC-2.0*SUM)/(2.0*A)
        BETA=B/A
        QLOWER=INT(ALPHA-SQRT(ALPHA**2-BETA))
        QUPPER=INT(ALPHA+SQRT(ALPHA**2-BETA))
        CALL LCMN(XNEW,NOS,LCM)
        T=QLOWER;U=LCM
        V=T/U+0.5
        QLOWER=LCM*INT(V)
        T=QUPPER;U=LCM
        V=T/U+0.5
        QUPPER=LCM*INT(V)
        IF(QLOWER.EQ.0) QLOWER=LCM
        IF(QUPPER.EQ.0) QUPPER=LCM
        QNEW=QLOWER
        DO 70 I=1,NOS
        XNEW(I)=INT(SQRT(BX(I)/AX(I)+0.25)+0.5)
        T=QNEW;U=XNEW(I)
        V=T/U+0.5
        YNEW(I)=INT(V)
        IF(YNEW(I).EQ.0)YNEW(I)=1
70      CONTINUE
```

```
 150  X=[illegible](A,XX,YNEW)
      Z=[illegible](0,[illegible],Y[illegible])
      [illegible]=INT(SQRT(LX/X+0.25)+0.5)
      CALL L[illegible](T[illegible],NOS,LCM)
      C=[illegible]
      [illegible]=A[illegible]
      [illegible]
      IF([illegible])[illegible]=LCM
      IF([illegible].GT.[illegible]) GO TO 500
      DO 67 I=1,NOS
      X[illegible](I)=[illegible]/Y[illegible]
 67   CONTINUE
      PSUM=0.;QSUM=0.
      DO 66 I=1,NOS
      PSUM=PSUM+AX(I)*XNEW(I)
      QSUM=QSUM+BX(I)/XNEW(I)
 66   CONTINUE
      CVR=A[illegible]+B/[illegible]+PSUM+QSUM
      IF(CVR.GT.CMIN)GO TO 400
      CMIN=CVR;QOPT=QNEW
      DO 75 I=1,NOS
      YMIN(I)=YNEW(I);XMIN(I)=XNEW(I)
 75   CONTINUE
 400  DO 76 I=1,NOS
      IX(I)=YNEW(I)
 76   CONTINUE
      MIN=999999
      DO 80 I=1,NOS
      JTL=IY(I)
      JTLL=JTL+1
      QAUX(I)=[illegible](QX(I)/AY(I)*JTL*JTLL)
      IX(I)=JTLL
      CALL LECOM(IX,NOS,LCM)
      T=QAUX(I);U=LCM
      V=T/U+0.5
      QAUX(I)=LCM*INT(V)
      IF(MIN.LT.QAUX(I)) GO TO 80
      MIN=QAUX(I);J1=I
 80   CONTINUE
      IF(MIN.GE.QUPPER) GO TO 500
      YNEW(J1)=YNEW(J1)+1
      GO TO 150
 500  COPT=CMIN;QOPT=QMIN
      DO 120 I=1,NOS
```

```
            YOPT(I)=YAUX(I)
            XOPT(I)=XAUX(I)
      120   CONTINUE
            CALL [illegible](T1,T2)
            [illegible]
            WRITE(21,31)
      31    FORMAT(1X,'OPTIMUM RESULTS')
            WRITE(21,*),COPT,TOPT,(YOPT(I),I=1,NOS),(XOPT(I),I=1,NOS)
            WRITE(21,121),TIME
      121   FORMAT(1X,'CPU TIME TAKEN: ',F5)
            STOP
            END

      C     *****************************************************************

      C     THIS FUNCTION CALCULATES THE SUM OF ax(I) FOR ALL STAGES.

      C     *****************************************************************
            FUNCTION ASUMNG(C,CX,YAUX)
            INTEGER YAUX(20)
            DIMENSION CX(20)
            COMMON NOS
            SUM=0.0
            DO 10 I=1,NOS
            SUM=SUM+CX(I)/YAUX(I)
      10    CONTINUE
            ASUMNG=C+SUM
            RETURN
            END

      C     *****************************************************************

      C     THIS FUNCTION CALCULATES THE SUM OF b/x(I) FOR ALL STAGES.

      C     *****************************************************************
            FUNCTION BSUMNG(C,CX,YAUX)
            INTEGER YAUX(20)
            DIMENSION CX(20)
            COMMON NOS
            SUM=0.0
            DO 10 I=1,NOS
            SUM=SUM+CX(I)*YAUX(I)
      10    CONTINUE
            BSUMNG=C+SUM
            [illegible]
      END

C     ****************************************************************

C     THIS SUBROUTINE CALCULATES THE LCM FOR THE GIVEN VECTOR.

C     ****************************************************************
      SUBROUTINE [illegible](Y,NOS,LCM)
      INTEGER R1,R2,Y(20),YY(20)
      DIMENSION LF(40)
      LOGICAL FLAG
      IF(NOS.NE.1) GO TO 1
      LCM=Y(NOS)
      RETURN
1     DO 2 I=1,NOS
      YY(I)=Y(I)
2     CONTINUE
      [illegible]=Y(NOS)
      K=1
5     FLAG=.FALSE.
      L=1
6     IA=MAX0(Y(K),Y(K+1))
      DO 10 I=2,IA
      R1=MOD(Y(K),I)
      R2=MOD(Y(K+1),I)
      IF((R1.EQ.0).AND.(R2.EQ.0)) GO TO 15
      IF((.NOT.(FLAG)).AND.(R1.NE.0).AND.(R2.NE.0)) GO TO 25
      FLAG=.TRUE.
      GO TO 10
25    FLAG=.TRUE.
10    CONTINUE
      GO TO 40
15    LF(L)=I;L=L+1
      Y(K)=Y(K)/I;Y(K+1)=Y(K+1)/I
      GO TO 6
40    IF(.NOT.(FLAG)) GO TO 50
      LF(L)=1
      GO TO 60
50    LF(L)=Y(K)*Y(K+1)
60    LCM=1
      DO 20 I=1,L
      LCM=LCM*LF(I)
20    CONTINUE
      K=K+1;Y(K)=LCM
```

```
0268              IF(I.NE.[illegible]) GO TO 79
0269              GO TO 9
0270     78       Y(I,J)=[illegible]
0271              DO 11 I=1,[illegible]
0272              Y(I)=X(I)
0273     11       CONTINUE
0274              RETURN
0275              END
```

```
C         *************************************************************


C         THIS PROGRAM IS FOR VARIABLE [illegible] MODEL.
C         THE INPUTS ARE TO BE GIVEN IN THE FOLLOWING ORDER.DEMAND,
C         NO. OF STAGES,SETUP COSTS,INVENTORY CARRYING COSTS,
C         PRODUCTION RATES.THE OUTPUT GIVES THE OPTIMAL COST,
C         OPTIMAL LOTSIZE & INTEGER RATIOS.

C         [illegible] OF [illegible] STAGES ALLOWED.

C         *************************************************************
          DIMENSION P(20),SETUP(20),AK(20),IK(20),KL(20,20)
          REAL NSUM,INCOST(20)
          INTEGER K(20),D
          LOGICAL [illegible]
          READ(11,*),D,NOS
          READ(11,*),(SETUP(I),I=1,NOS)
          READ(11,*),(INCOST(I),I=1,NOS)
          READ(11,*),(PR(I),I=1,NOS)
          N=NOS!ACOST=7.0E+11
          PN=[illegible]K(N)=1
          AK(N)=FLOAT(K(N))
          X=FLOAT(D)/FLOAT(PN)
          QN=SQRT(2.0*D*SETUP(NOS)/(INCOST(N)*(1.0-X)))
          X=FLOAT(D)/FLOAT(PN)
          TC=SQRT(2.0*D*SETUP(NOS)*INCOST(N)*(1.0-X))
          TYPE *,QN,TC
          N=N-1
          X=FLOAT(D)/FLOAT(PN)
   90     NSUM=SETUP(NOS)*D!DSUM=INCOST(NOS)*(1.0-X)
   100    AK(N)=SQRT(2.0*D*SETUP(N)/(QN**2*INCOST(N)*(1.0-X)))
          AK(N)=AMAX1(AK(N+1),AK(N))
          IF(AK(N).GT.AK(N+1)) GO TO 50
          AK(N)=AK(N+1)
   50     IK=AK(N)/AK(N+1)+0.5
          IF(IK.LT.1)IK=1
          K(N)=K(N+1)*IK
   60     NSUM=NSUM+D*SETUP(N)/K(N)
          X=FLOAT(D)/FLOAT(PN)
          DSUM=DSUM+K(N)*INCOST(N)*(1.0-X)
          TYPE *,QN,K(N)
          QN=SQRT(2.0*NSUM/DSUM)
          CUR=SQRT(2.0*NSUM*DSUM)
```

```
      [illegible]
      IF(I.GT.[illegible]) GO TO [illegible]
      GO TO [illegible]
200   READ(43,*) COST,DN,(K(I),I=1,NOS)
      CALL [illegible](A,B,NOS,ISTOP,IACOST,PR,KK,JJ)
      DO 40 I=1,NOS
      K(I)=KK(I,JJ)
40    CONTINUE
      TYPE *,(K(I),I=1,NOS)
      STOP
      CALL [illegible](NOS,ACP)
      TYPE *,ACP,K(I)
      [illegible]
      [illegible]
      I=NOS+1
      TYPE *,IS,[illegible]
      ITYPA=ITYPE(I)
      TYPE *,ITYPA
      IF(ITYPA.GT.1) GO TO 110
      GO TO 190
110   MATCH=.TRUE.
      DO 20 I=1,NOS
      IF(MATCH.AND.(KK(I).EQ.K(I))) GO TO 20
      MATCH=.FALSE.
20    CONTINUE
190   ASUM=0.0;SUM=0.0
      DO 40 I=1,NOS
      SUM=SUM+IRSTYP(I)/(K(I)*DN)
      X=SQRT(S)/FLOAT(PR)
      ASUM=ASUM+0.5*IRCOST(I)*DN*K(I)*(1.0-X)
      COST=SUM+ASUM
40    CONTINUE
      TYPE *,COST
      IF(ITYPA.EQ.1) GO TO 310
      IF(MATCH) GO TO 300
310   IF(COST.GT.ACOST) GO TO 300
      ACOST=COST
      DO 30 I=1,NOS
      KK(I)=K(I)
30    CONTINUE
      I=NOS-1
      GO TO 90
300   WRITE(43,*),COST,(K(I),I=1,NOS),DN
      STOP
```

```
              END
      C       ****************************************************

      C       THIS SUBROUTINE CALCULATES THE LCM FOR THE GIVEN VECTOR.
      C
      C       ****************************************************
              SUBROUTINE LCMF(Y,NOS,LCM)
              INTEGER R1,R2,Y(25),YY(25)
              DIMENSION LF(250)
              LOGICAL FLAG
              IF(NOS.NE.1) GO TO 1
              LCM=Y(NOS)
              RETURN
      1       DO 2 I=1,NOS
              YY(I)=Y(I)
      2       CONTINUE
              LIM=Y(NOS)
              K=1
      5       FLAG=.FALSE.
              L=1
      6       IM=MIN0(Y(K),Y(K+1))
              DO 10 I=2,IM
              R1=MOD(Y(K),I)
              R2=MOD(Y(K+1),I)
              IF((R1.EQ.0).AND.(R2.EQ.0)) GO TO 15
              IF((.NOT.(FLAG)).AND.(R1.NE.0).AND.(R2.NE.0)) GO TO 25
              FLAG=.FALSE.
              GO TO 10
      25      FLAG=.TRUE.
      10      CONTINUE
              GO TO 40
      15      LF(L)=I;L=L+1
              Y(K)=Y(K)/I;Y(K+1)=Y(K+1)/I
              GO TO 6
      40      IF(.NOT.(FLAG)) GO TO 50
              LF(L)=1
              GO TO 60
      50      LF(L)=Y(K)*Y(K+1)
      60      LCM=1
              DO 20 I=1,L
              LCM=LCM*LF(I)
      20      CONTINUE
              K=K+1;Y(K)=LCM
              IF(K.EQ.NOS) GO TO 70
              GO TO 5
```

```
70    Y(NOS)=IORD
      DO 11 I=1,NOS
      Y(I)=YY(I)
11    CONTINUE
      RETURN
      END

C     ************************************************************

C     THIS SUBROUTINE SEARCHES FOR OPTIMUM INTEGER RATIO FOR
C     THE BATCHES FOR THE GIVEN STAGES.

C     ************************************************************
      SUBROUTINE SEARCH(L,D,NOS,SETUP,INCOST,PN,KK,JJ)
      REAL INCOST(20)
      INTEGER PN(20),D
      DIMENSION KS(20,20),K(20),COSTI(20),IK(20),KI(20),SETUP(20)
      N=NOS;COSTI(N+1)=7.E+16
      K(N+1)=L
      TYPE *,(K(I),I=1,NOS)
5     ACOST=7.E+16
      DO 7 L=K(N+1),1,-1
      DO 17 I=K(L),1,-1
      DO 15 J=1,NOS
      IF(K(J).GE.I) GO TO 17
      KI(J)=K(J)
      GO TO 15
17    IK(J)=FLOAT(K(J))/FLOAT(I)
      IF(IK(J).EQ.0)IK(J)=1
      KI(J)=IK(J)*I
      KI(J)=I
15    CONTINUE
      SUM=0.0;ASUM=0.0
      DO 20 II=1,NOS
      SUM=SUM+SETUP(II)/KI(II)
      ASUM=ASUM+KI(II)*INCOST(II)*(1.0+FLOAT(D)/FLOAT(PN))
20    CONTINUE
      COST=SQRT(2.0*D*SUM*ASUM)
      IF(K(N).EQ.1) GO TO 22
      IF(COST.GT.ACOST) GO TO 30
22    ACOST=COST
      DO 25 LI=1,NOS
      KS(LI,N)=KI(LI)
25    CONTINUE
```

```
30    [illegible]=[illegible]
10    CONTINUE
7     CONTINUE
      IF(POSIT(I).LT.POSIT(I+1)) JJ=[illegible]
      WRITE(61,[illegible]),[illegible],[illegible],JJ,(KA(I,[illegible]),I=1,NOS)
      [illegible]
      IF([illegible]) GO TO [illegible]
      GO TO [illegible]
40    [illegible]
      DO 45 I=1,[illegible]
      [illegible]
      [illegible]=[illegible]*KA(I,JJ)*[illegible](I)*(1.0-FLOAT(D)/FLOAT(PS))
45    CONTINUE
      COSTS=[illegible]
      WRITE(61,[illegible]),(KA(I,JJ),I=1,NOS),COSTS
      RETURN
      END
      C     ******************************************************************
      C     THIS PROGRAM IS FOR VARIABLE LOT SIZE MODEL WITH LOT SPLITTING
      C
      C     ******************************************************************
            DIMENSION P(0:20),D(0:20),[illegible],TI(0:20),G(0:20),R(0:20)
            [illegible] (0:20),YC(0:20),VV(20),S(20)
            [illegible] XL(0:20),QL(0:20),A(0:20),B(0:20),H(0:20),SC(0:20)
            DIMENSION IS(0:20) ,TP(0:20),Q(0:20),T(0:20),F(0:20)
            REAL L(0:20)
            INTEGER DELTA(0:20),ISUM(0:20),V(20)
            INTEGER ITMP1,ITMP2,ITMP
            READ (21,*),N,D
            READ (21,*),(P(I),F(I),TI(I),CI(I),I=1,N)
            READ(22,*),(L(I),G(I),I=1,N)
            Z=1.0;ITERA=1
            P(N+1)=0.;TOTCST=1.0E+29
            P(0)=D;CI(N+1)=0.0
            QL(0)=QL(1);IS(0)=1
            ISUM(0)=IS(0)=1
            CALL [illegible](TI,F1)
            DO 10 I=1,N
            DELTA(I)=0
            IF(P(I).LT.P(I+1)) DELTA(I)=1
      10    CONTINUE
            DO 12 I=1,N
            IF(I.NE.N) GO TO 11
            Z=0.0
            GO TO 13
      11    Z=1.0/P(I+1)
      13    VV(I)=CI(I)*DELTA(I+1)*(Z-1.0/P(I))
            B(I)=((1.0/D-1.0/P(I))*(CI(I)-CI(I+1)))/2.0+CI(I)*DELTA(I+1)
            1 1.0/P(I))
            S(I)=B(I)-VV(I)
            H(I)=CI(I)*((1.0-DELTA(I))/P(I)+DELTA(I)/P(I-1))
      12    CONTINUE
            CALL [illegible](N,L,G,R,P,D,TI,QL,XL)
            QL(0)=QL(1)
            CALL [illegible]CAP(N,D,XL,QL,RCAP)
            ISUM(0)=1
            DO 20 I=1,N
            IS(I)=[illegible](QL(I+1)/(QL(I)*ISUM(I-1)))
      IF(IS(I).EQ.0)IS(I)=1
      ISUM(I)=ISUM(I-1)+IS(I)
 20   CONTINUE
 15   A[illegible]=7.[illegible]
      DO 30 I=2,N
      A=[illegible](I)/[illegible](I-1)
      IF(A.LT.A[illegible]) [illegible]=A
 30   CONTINUE
      QL(1)=A[illegible](
[illegible],[illegible](1))
      [illegible] [illegible],QL(1),[illegible],QL(1)
      DO 22 I=2,N
      QL(I)=QL(1)*[illegible](I-1)
 22   CONTINUE
      DO 50 I=2,N
      IF(COSTS(I).EQ.1) GO TO 100
      IS(I)=IFIX(SQRT((QL(I)*4?*CI(I))/(P(I)*PI(I))+0.25)+0.5)
      NN=IFIX(QL(I)/S(I)+0.9999999)
      IS(I)=MAX0(IS(I),NN)
      XL(I)=QL(I)/IS(I)
      IR(I)=QL(I)/XL(I)
      GO TO 50
 100  H(I)=((QL(I)/IS(I))*(CI(I)/P(I-1)))
      V(I)=[illegible](I)*IS(I))/QN(I))
      Z(I)=(QL(I)/IS(I))*(1.0/P(I)-1.0/P(I-1))*CI(I)
      YC(I)=SQRT(H(I)/V(I))
      K=(INT(YC(I)+0.9999))*IS(I)
      XKIN=(QL(I)/G(I))
      IF(K.LE.XKIN) GO TO 80
      KK=(INT(YC(I)))*IS(I)
      IF(KK.EQ.0)KK=1
      IF(K.EQ.0)K=1
      TOC=1.0E+20
      DO 60 LL=KK,K,1
      IJK=LL
      IKL=I
      CALL RIP(P,XL,QL,IKL,IJK,D,R)
      COST=(F(I)+LL*FI(I))/QL(I)+0.5*CI(I)*(2.0*R(LL-1)+
     1  QL(I)*(1.0/D-1.0/P(I))-QL(I-1)*(1.0/D-1.0/P(I-1)))
      IF(COST.GT.TOC) GO TO 60
      TOC=COST
      J1=LL
 60   CONTINUE
      TC=TC+TOC
      IO(I)=J1
      [illegible]
   80 K=IFIX(QL(I)/Q(I)+0.99999)
      KK=IFIX(FLOAT(K)/FLOAT(I3(I))+0.99999)*I3(I)
      TOC=1.0E+29
      DO 70 LL=K,KK,1
      II=LL
      JJ=1
      CALL RET(P,XL,QL,IKL,IJK,D,R)
      COST=(F(I)+LL*TI(I))/QL(I)+0.5*CI(I)*(2.0*R(LL-1)+
     1 QL(I)*(1.0/D-1.0/P(I))-QL(I-1)*(1.0/D-1.0/P(I-1)))
      IF(COST.GT.TOC) GO TO 70
      TOC=COST
      JJ=LL
   70 CONTINUE
      I3(I)=JJ
      TC=TC+TOC
  110 X6(I)=QL(I)/I3(I)
      II=I3(I)
      II=I
      CALL RCAP(L,D,XL,QL,RCAP)
      CALL RET(P,XL,QL,II,KI,D,R)
      IF(ABS(R(I-1)-RCAP(I-1)).GT.0.00005) GO TO 50
      CALL RIV(P,XL,QL,II,KI,D,R)
      RR=R(I-1)-XL(I)/P(I)
      PRODT=CI(I)*TI(I)/QL(I)+(CI(I)*RR)
      RX=PRODT**2-4.0*TI(I)*CI(I)/(QL(I)**2*P(I))
      K=IFIX((PRODT+SQRT(RX))/(2.0*TI(I)/QL(I))+0.5)
      KK=IFIX((PRODT-SQRT(RX))/(2.0*TI(I)/QL(I))+0.5)
      IF(K.EQ.0)K=1
      IF(KK.EQ.0)KK=1
      IF((K.EQ.1).AND.(KK.EQ.1))GO TO 230
      I3(I)=1
      GO TO 50
  230 TOC=1.0E+29
      DO 120 LL=KK,K
      COST=(F(I)+LL*TI(I))/QL(I)+0.5*CI(I)*(2.0*R(LL-1)+
     1 QL(I)*(1.0/D-1.0/P(I))-QL(I-1)*(1.0/D-1.0/P(I-1)))
      IF(COST.GT.TOC) GO TO 120
      TOC=COST
      JJ=LL
  120 CONTINUE
      I3(I)=JJ
      X6(I)=QL(I)/I3(I)
      TC=TC+TOC
   50 CONTINUE
```

```
      [illegible]=0.0
      DO 130 I=1,N
      IF(I.EQ.N) GO TO 131
      Z=1.0/P(I+1)
131   IF(I.EQ.N)Z=0.0
      [illegible](T(I)+TR(I)*Q(I))/TSUM(I-1)
      PP=(SUM(I-1)*(1.0/D-1.0/P(I))+(X(I)-X(I+1))/2.0)
      QI=(SUM(I-1)*DELTA(I+1)*(Z-1.0/P(I))+(1.0-DELTA(I))/
     X (P(I)*DA(I)+DELTA(I)/(P(I-1)[illegible](I))))*CT(I)
      PO=PO+PP
130   CONTINUE
      QCOM=[illegible]
      DO 310 I=1,N
      IR(I)=QL(I)/AL(I)
      XL(I)=QL(I)/IR(I)
310   CONTINUE
      DO 410 I=1,N
      II=I
      QL=IR(II)
      CALL RIV(P,XL,QL,TT,GL,N,R)
410   CONTINUE
      ASUM=0.0;TTCST=0.0
      DO 510 I=1,N
      Z1=(P(I)+TR(I)*T(I))/QL(I)
      Z2=0.5*CT(I)*(2.0*R(I-1)+QL(I)*(1.0/D-1.0/P(I))-
     XQL(I-1)*(1.0/D-1.0/P(I-1)))
      ASUM=ASUM+(Z1+Z2)
510   CONTINUE
      TTCST=ASUM*D
      IF(TTCST.GE.TOTCST) GO TO 200
      TOTCST=TTCST
520   WRITE(23,260),TOTCST,ITERA
260   FORMAT(2X,'THE TOTAL COST = ',F10.4,2X,'ITERATION NO: ',I3)
      IF(ABS(QCOM-QL(1)).LT.0.0005) GO TO 200
      QL(1)=QCOM
      TYPE 51,ITERA
51    FORMAT(2X,I2,2X,'ITERATION')
      ITERA=ITERA+1
      IF(ITERA.EQ.20) GO TO 200
      GO TO 15
200   CALL LCOM(IR,N,LCNM)
      TYPE *,[illegible],(IR(I),I=1,N)
      DO 2110 I=1,N
      X(I)=IR(I)
```

```
210     CONTINUE
        [illegible]
        CALL [illegible](?,IOE,ICT)
        [illegible]
        IXX=INT(QL(1)/UCH+0.5)
        QL(1)=IXX*UCH
        DO 2100 J=2,N
        QL(J)=QL(J-1)*TS(J)
2100    CONTINUE
        CALL [illegible](ETIME2)
        ITIME=ETIME2-ETIME1
        WRITE(22,261),(QL(I),I=1,N)
261     FORMAT(4(3X,F9.4,3X))
        WRITE(23,262),(XL(I),I=1,N)
262     FORMAT(4(3X,F9.4,3X))
        WRITE(22,263),(TS(I),I=1,N)
263     FORMAT(4(3X,I3,3X))
        WRITE(23,263),(TS(I),I=1,N)
        WRITE(23,881),ITIME
881     FORMAT(4X,'THE CPU TIME TAKEN IS: ',I6)
        WRITE(21,265 ),(QL(I),IS(I),I=1,N)
265     FORMAT(1X,'LOTSIZE: ',F9.4,2X,'NO.OF BATCHES: ',I3)
        WRITE(23,263),(IS(I),I=1,N),(TS(I),I=1,N)
        STOP
        END

C       THIS ROUTINE FINDS THE SOLUTION TO THE RELAXED CONSTRAINT PR
C
C       ************************************************************
        SUBROUTINE QLR(N,L,G,B,A,H,GG,QL,XL)
        DIMENSION G(0:20),A(0:20),B(0:20),GG(0:20),H(0:20),QL(0:20)
        DIMENSION XL(0:20)
        REAL L(0:20)
        QL(0)=0.0
        DO 10 I=1,N
        QL(I)=INT(SQRT(ABS(A(I)/B(I)))+0.25))
        IF(I.EQ.1) GO TO 30
        AMIN=QL(I-1)
        GO TO 30
20      AMIN=7.0E+1
30      QL(I)=AMIN1(QL(I),L(I))
        QL(I)=AMAX1(QL(I),QL(I-1))
        QL(I)=AMIN1(QL(I),L(I))
        XL(I)=SQRT(GG(I)/H(I))
```

```
   10  CONTINUE
       RETURN
       END
       SUBROUTINE [illegible](C,XL,QL,I,LL,Q,R)
       DIMENSION C(0:20),XL(0:20),QL(0:20),P(0:20),THETA(0:20)
       DO 10 II=1,LL
       I=II-1
       IF(LL.EQ.0)[illegible]
       X=INT((I*XL(I))/(LL*P(I-1)))*QL(I-1)*(1.0/P-1/P(I-1))
       C(I)=I*QL(I)*(1.0/P(I)-1.0/P(I-1))/LL-X
   10  CONTINUE
       R(I-1)=XL(I)/P(I)+MAXIM(THETA,LL)
       RETURN
       END
       FUNCTION MAXIM(THETA,I)
       DIMENSION THETA(0:20)
       AMIN=0.0
       DO 10 II=1,I
       I1=II-1
       IF(THETA(I1).GT.AMIN) AMIN=THETA(I1)
   10  CONTINUE
       MAXIM=AMIN
       RETURN
       END
       SUBROUTINE RCAP(N,P,XL,QL,RCAP)
       DIMENSION P(0:20),XL(0:20),QL(0:20),RCAP(0:20),L(0:20)
       DO 10 I=1,N
       IF(P(I).GE.P(I-1)) L(I)=1
       IF(P(I).LT.P(I-1)) GO TO 20
       GO TO 10
   20  L(I)=INT(QL(I-1)/XL(I)+0.99999)
   10  CONTINUE
       DO 40 I=1,N
       RCAP(I-1)=XL(I)*(1.0/P(I)+(L(I)-1)*(1.0/P(I)-1.0/P(I-1)))
   40  CONTINUE
       RETURN
       END

C       *******************************************************************
C
C       THIS ROUTINE FINDS THE LCM FOR THE GIVEN VECTOR.
C
C       *******************************************************************
```

```
      SUBROUTINE LLCM(Y,NOS,LCM)
      INTEGER Y1,Y0,Y(29),YY(29)
      DIMENSION LF(730)
      LOGICAL FLAG
      IF(NOS.NE.1) GO TO 1
      LCM=Y(NOS)
      RETURN
 1    DO 2 I=1,NOS
      YY(I)=Y(I)
 2    CONTINUE
      TYPE 7,NOS,(Y(I),I=1,NOS)
      IBIG=Y(NOS)
      K=1
 5    FLAG=.FALSE.
      L=1
 6    IM=MAX0(Y(K),Y(K+1))
      DO 10 I=2,IM
      R1=MOD(Y(K),I)
      R2=MOD(Y(K+1),I)
      IF((R1.EQ.0).AND.(R2.EQ.0)) GO TO 15
      IF((.NOT.(FLAG)).AND.(R1.NE.0).AND.(R2.NE.0)) GO TO 25
      FLAG=.FALSE.
      GO TO 10
 25   FLAG=.TRUE.
 10   CONTINUE
      GO TO 40
 15   LF(L)=I;L=L+1
      Y(K)=Y(K)/I;Y(K+1)=Y(K+1)/I
      GO TO 6
 40   IF(.NOT.(FLAG)) GO TO 50
      LF(L)=1
      GO TO 60
 50   LF(L)=Y(K)*Y(K+1)
 60   LCM=1
      DO 20 I=1,L
      LCM=LCM*LF(I)
 20   CONTINUE
      K=K+1;Y(K)=LCM
      IF(K.EQ.NOS) GO TO 70
      GO TO 5
 70   Y(NOS)=IBIG
      DO 11 I=1,NOS
      Y(I)=YY(I)
 11   CONTINUE
```